Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Eletrônica

Proposta de uma Arquitetura de Redes Neurais para Estimativa da Frequência Cardíaca Fetal a Partir do ECG Abdominal em Gestantes

Autor: Humberto Domingos de Carvalho Júnior
Orientador: Prof. Dr. Gilmar Silva Beserra

Brasília, DF
2018

Humberto Domingos de Carvalho Júnior

Proposta de uma Arquitetura de Redes Neurais para Estimativa da Frequência Cardíaca Fetal a Partir do ECG Abdominal em Gestantes

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Gilmar Silva Beserra

Brasília, DF

2018

Humberto Domingos de Carvalho Júnior
Proposta de uma Arquitetura de Redes Neurais para Estimativa da Frequência Cardíaca Fetal a Partir do ECG Abdominal em Gestantes/ Humberto Domingos de Carvalho Júnior. – Brasília, DF, 2018-
65 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Gilmar Silva Beserra

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.

1. Redes Neurais. 2. FECG. I. Prof. Dr. Gilmar Silva Beserra. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Proposta de uma Arquitetura de Redes Neurais para Estimativa da Frequência Cardíaca Fetal a Partir do ECG Abdominal em Gestantes

CDU 02:141:005.6

Humberto Domingos de Carvalho Júnior

Proposta de uma Arquitetura de Redes Neurais para Estimativa da Frequência Cardíaca Fetal a Partir do ECG Abdominal em Gestantes

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 17 de Abril de 2018:

Prof. Dr. Gilmar Silva Beserra
Orientador

Prof. Dr. Daniel Mauricio Muñoz Arboleda
Convidado

Prof. Dr. Daniel Chaves Café
Convidado

Brasília, DF
2018

"Não vos conformeis com este mundo,
mas transformai-vos pela renovação do vosso entendimento,
para que experimenteis qual seja a boa, agradável e perfeita vontade de Deus".
(Bíblia Sagrada, Romanos 12, 2)

Resumo

A estimativa da frequência cardíaca fetal (FHR – *Fetal Heart Rate*) tem se mostrado um parâmetro de fundamental importância na avaliação das condições do feto durante a gestação. A partir do ECG abdominal (AECG) da mãe, é possível estimar a FHR após o devido processamento do sinal original. Considerando que o AECG é composto pelo ECG da mãe, pelo ECG do feto e por ruídos diversos, várias abordagens têm sido utilizadas para extrair o ECG do feto (FECG) e, a partir dele, estimar a FHR contando os picos. Alguns exemplos de soluções propostas envolvem o uso de filtros adaptativos, *wavelets*, *blind source separation*, etc. Recentemente, alguns trabalhos de conclusão de curso na Engenharia Eletrônica da Faculdade do Gama – UnB têm sido direcionados para a implementação de um protótipo para estimar a FHR utilizando um FPGA na parte de processamento, pois o mesmo permite fazer aceleração de algoritmos e também utilizar diferentes abordagens, visto que é reconfigurável. Além do bloco de processamento, no qual está sendo utilizada atualmente uma abordagem que usa um filtro adaptativo e o método dos mínimos quadrados (LMS), o protótipo também contém blocos de aquisição de sinal e comunicação. Considerando que o filtro adaptativo apresentou um desempenho aceitável com sinais simulados de ECG, mas não apresentou bons resultados com sinais provindos de bases de dados, este trabalho tem como objetivo a proposta de uma arquitetura de redes neurais para o bloco de processamento do protótipo. Espera-se, assim, obter um melhor resultado, visto que que as redes neurais são adaptativas às características não lineares e variantes no tempo dos sinais de ECG.

Palavras-chaves: ECG. ECG fetal. ECG Abdominal. Engenharia Biomédica. Redes Neurais. Redes Neurais Convolucionais. Processamento de Sinais.

Abstract

Fetal heart rate (FHR) has been a fundamental parameter in the evaluation of the fetal condition during gestation. Starting from the mother's abdominal ECG, it's possible to estimate the FHR through adequate signal processing. Considering that this signal is composed by the mother ECG, fetal ECG and noise, a variety of forms has been used to extract the fetal ECG (FECG) and, from this signal, estimate the FHR counting the peaks. Some examples of solution cover the use of adaptive filtering, neural networks, wavelets, blind source separation, etc. Recently, some final year projects in Electronic Engineering at Faculdade do Gama - UnB have been directed to the implementation of a prototype to estimate the FHR using an FPGA as processing unit, because it allows us to accelerate algorithms and also make use of different approaches, since its reconfigurable. Beyond the processing unit, that has been developed using an adaptive filter and least minimum square algorithm, the prototype also has signal acquisition and communication blocks. Considering that the adaptative filtering presented an acceptable performance using simulated ECG signal, but it didn't bring out good results using database signals, this work aims to propose a neural network architecture to be used in the processing unit of the prototype. It is expected to obtain a better result, since neural networks are adaptive to the nonlinear and variant characteristics of the ECG signal.

Key-words: ECG. fetal ECG. Abdominal ECG. Biomedic Enginnering. Neural Networks. Convolutional Neural Networks Signal Processing.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AECG	Eletrocardiograma Abdominal
ACOG	*American Congress of Obstetricians and Gynecologists*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
ASIC	*Application-Specific Integrated Circuit*
ASSP	*Application-Specific Standard Product*
BPM	Batimento por minuto
BRAM	*Block Random Access Memory*
CLB	*Configurable Logic Block*
CNN	*Convolutional Neural Network*
ECG	Eletrocardiograma
EMG	Eletromiograma
FECG	Eletrocardiograma Fetal
FHR	*Fetal Heart Rate*
FPGA	*Field-Programmable Gate Array*
IDE	*Integrated Design Environment*
LMS	*Least Minumum Square*
LUT	*Look-up Table*
MECG	Eletrocardiograma Materno
PLL	*Phase-Locked Loop*
RAM	*Random Access Memory*
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Network*

RISC	*Reduced Instruction Set Computer*
SoC	*System on a chip*
SVD	*Singular Value Decomposition*

Sumário

1 Introdução

1.1 Contextualização

A monitorização dos sinais biológicos do paciente é a principal fonte de informação da clínica médica para obtenção de diagnósticos. No que diz respeito à gestação, essa monitorização é ainda mais crítica, tendo em vista que não existem formas visíveis de se conhecer a saúde fetal. Segundo a ACOG (*American Congress of Obstetricians and Gynecologists*) as doenças cardíacas congênitas são a principal causa de morte infantil e representam 24% de todos os registros de óbitos de bebê (MURPHY; KOCHANEK; XU, 2015). Essas doenças, na maioria dos casos, se tratadas ainda nos primeiros meses da gestação, oferecem menor risco à saúde da criança. Ainda segundo a ACOG, a cada 1000 nascimentos, sete bebês sofrem com falta de oxigenação durante o parto, o que pode causar danos cerebrais ou até mesmo a morte (OBSTETRICIANS; GYNECOLOGISTS et al., 2015).

Como uma forma de identificação de doenças e má formação, o monitoramento da FHR (*Fetal Heart Rate*), ou frequência cardíaca fetal, é a principal forma de se obter um acompanhamento adequado do desenvolvimento fisiológico do feto. Os dados obtidos por esse indicativo podem levar à redução dos casos de lesão cerebral e das taxas de mortalidade de recém-nascidos. Estudos realizados com base em vários casos de danos cerebrais ocasionados durante o parto mostram que cerca de 50% das lesões poderiam ser evitadas. A maioria dos erros está relacionada à falha humana, principalmente no que se diz respeito à falha na interpretação ou utilização dos dados da FHR. (WARRICK; HAMILTON; MACIESZCZAK, 2005)

Atualmente, o eletrocardiograma fetal (FECG) é uma das formas de obtenção da FHR. Porém, a extração precisa desse sinal ainda é um desafio. Acredita-se que, com melhorias nas técnicas de obtenção do sinal, é possível que ele forneça dados que vão permitir outros diagnósticos, além dos que já são fornecidos levando em consideração apenas a frequência cardíaca (FREEMAN et al., 2012).

A primeira vez que um sinal de FHR foi obtido por meios eletrônicos foi em 1906, por CREMER, e foi usado somente para o diagnosticar se o bebê estava vivo ou não. Vários outros médicos e cientistas, depois de M. Cremer, tentaram, sem sucesso, obter diagnósticos a partir da frequência cardíaca fetal, até que em 1960, HON publicou sobre o primeiro aparelho (Figura 1) capaz de realizar o monitoramento contínuo e instantâneo da FHR através de sinais obtidos do abdômen materno. Estudos posteriores de HON e outros colegas levaram ao reconhecimento de padrões da FHR que são relacionados à

bradicardia, à taquicardia e à atividade fetal.

Figura 1 – O primeiro monitor fetal disponível para comercialização. Fonte: (FREEMAN et al., 2012)

Desde então, várias técnicas foram desenvolvidas e têm sido aprimoradas para a obtenção do sinal de FECG, alguns exemplos de soluções propostas que se destacam envolvem o uso de filtros adaptativos, *wavelets*, *blind source separation*, redes neurais e decomposição em valores singulares (SVD - do inglês *Singular-Value Decomposition*).

Recentemente, alguns trabalhos de conclusão de curso na Engenharia Eletrônica da Faculdade do Gama – UnB têm sido direcionados para a implementação de um protótipo para estimar a FHR de forma não invasiva utilizando um FPGA na parte de processamento, pois o mesmo permite fazer aceleração de algoritmos e também utilizar diferentes abordagens, visto que é reconfigurável. O projeto é composto pelo bloco de processamento, no qual está sendo utilizada atualmente uma abordagem que usa um filtro adaptativo e o método dos mínimos quadrados (LMS), e também blocos de aquisição de sinal e comunicação.

1.2 Objetivos

1.2.1 Objetivo Geral

Propor uma arquitetura de rede neural artificial para um módulo estimador da frequência cardíaca fetal a partir do sinal do ECG abdominal em gestantes.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Propor uma arquitetura de rede neural viável para realizar a detecção do pico da FECG.
- Teste baseado em sinais simulados
- Desenvolver código em C correspondente à arquitetura proposta;
- Estimar o desempenho do sistema em comparação com outras soluções propostas

1.3 Estrutura do Trabalho

O presente trabalho está divido em 6 partes. O Capítulo 1 contém a introdução, contextualização e objetivos do trabalho. No Capítulo 2, são apresentados os fundamentos teóricos, iniciando com uma explanação breve sobre o desenvolvimento cardíaco fetal e evoluindo para os conceitos de *software* em que se baseia a proposta de solução. No Capítulo 3, é abordada a metodologia do projeto, como também as ferramentas utilizadas para o desenvolvimento do mesmo. O Capítulo 4 apresenta a forma como a rede foi proposta e implementada, por meio de um exemplo que mostra a forma como a rede trata determinado conjunto de dados. Os resultados obtidos e as discussões sobre o projeto são apresentados no Capítulo 5. Por fim, o Capítulo 6 contém a conclusão deste trabalho.

2 Fundamentos Teóricos

2.1 Frequência Cardíaca Fetal

2.1.1 Desenvolvimento e atividade cardíaca

O coração humano começa a sua atividade espontânea a partir do 21º dia da gestação, que é quando ocorre uma fase primária de desenvolvimento do miocárdio. Durante essa fase, o batimento se mantém baixo (cerca de 40 BPM) e a atividade cardíaca já pode ser observada. O sangue, que se concentra em uma cavidade inferior do coração, é ejetado para as artérias, de dentro para fora, assim como ocorre em uma bomba simples (CARLSON, 2013).

A partir do 35º dia, torna-se possível estudar a frequência cardíaca. Nessa fase de desenvolvimento, o batimento fica em torno de 100 BPM e o coração já tem sua anatomia bem definida (Figura 2). Os batimentos cardíacos aumentam consideravelmente a partir da 8ª semana, podendo chegar a até 160 BPM em condições normais, essa frequência permanece estável até a 15ª semana, quando a atividade cardíaca vai diminuindo gradativamente. O ritmo cardíaco fetal é sempre preciso durante o seu desenvolvimento, porém, próximo ao nascimento, as condições do útero, a posição do bebê e a sua movimentação podem levar a variações momentâneas da frequência cardíaca (CARLSON, 2013).

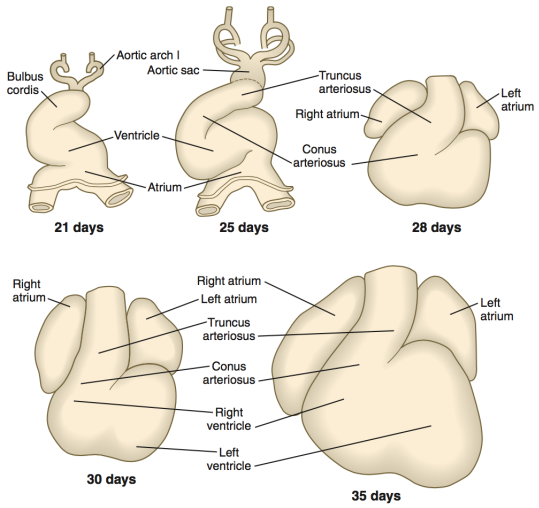

Figura 2 – Fases de desenvolvimento do coração. Fonte: (CARLSON, 2013)

2.1.2 FHR

O termo FHR *(Fetal Heart Rate)*, se refere ao intervalo entre as batidas consecutivas do coração fetal, em desenvolvimento, por unidade de tempo, ou seja, representa o batimento cardíaco e a atividade do coração. As informações proporcionadas pelo monitoramento da FHR são consideradas pela clínica médica de suma importância para conhecer as condições de saúde e desenvolvimento do feto, principalmente no que diz respeito à oxigenação adequada e à fisiologia do coração (FREEMAN et al., 2012). A Figura 3 indica como o exame é geralmente realizado hoje, com o auxílio do aparelho chamado cardiotocógrafo que obtêm a FHR através da ultrassonografia registrada pelo Efeito Doppler.

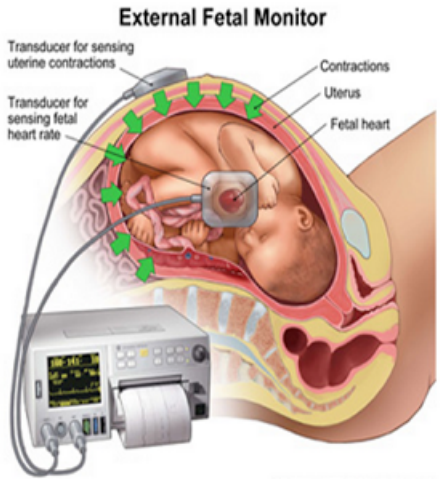

Figura 3 – Exame de cardiotocógrafia. Fonte: (DAVIES, 2012)

Quatro fatores são determinantes no acompanhamento da FHR: a média, a variação, a aceleração e a desaceleração do batimento. Desses quatro fatores, o mais importante é a média, porque todos os outros são determinados a partir dela. Embora exista um número de batimentos esperados para a FHR, que é entre 110-160 BPM, ela varia a cada fase da gestação, fazendo com que seja difícil se obter uma definição precisa da média (HATAI; CHAKRABARTI; BANERJEE, 2013).

É considerada uma aceleração do batimento fetal quando o mesmo se mantém por 15 segundos ou mais com +15 BPM em relação à média. Quando esse comportamento é associado à movimentação do feto, ele é considerado como um sinal de atividade do sistema nervoso central e bom desenvolvimento. De forma similar, a desaceleração do ritmo cardíaco ocorre quando o batimento se mantém por 15 segundos ou mais com -15 BPM em relação à média. Essa condição indica riscos à vida do bebê e está geralmente associada com a compressão do cordão umbilical, malformação do coração e bradicardia (HATAI; CHAKRABARTI; BANERJEE, 2013).

2.1.3 Eletrocardiograma Fetal (FECG)

Uma outra forma de se monitorar a FHR é através do Eletrocardiograma Fetal (FECG), que é o único parâmetro diagnosticável no começo da gestação. O sinal de FECG é muito similar ao ECG de um adulto pelo fato de que a atividade elétrica de um coração saudável é a mesma em todas as fases da vida. Então, as mesmas formas de onda que são vistas no ECG de um adulto podem ser observadas no FECG,a saber: onda P, complexo qRs e onda T, conforme mostra a Figura 4 (CLEMENTE et al., 2011).

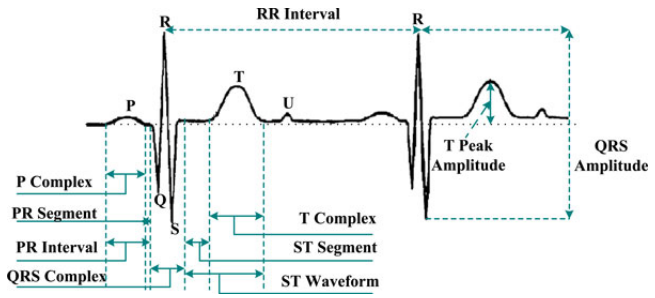

Figura 4 – Forma de onda obtida no ECG. Fonte:(HASAN; IBRAHIMY; REAZ, 2009)

É possível se obter o FECG de maneira não invasiva através de eletrodos posicionados no abdômen da mãe. Não é conhecida uma maneira ótima para o posicionamento dos eletrodos, tendo em vista que a posição do feto é variável. O sinal adquirido diretamente por eles é denominado AECG (Eletrocardiograma Abdominal) e, com o processamento adequado, o FECG pode ser extraído (HASAN; IBRAHIMY; REAZ, 2009).

O sinal do AECG é, contudo, constituído por vários sinais elétricos que dificultam a obtenção exata do FECG. O sinal predominante no AECG é o MECG (Eletrocardiograma Materno), advindo da própria atividade cardíaca da mãe. Além deste último citado, temos o EMG (Eletromiograma) advindo dos músculos do abdômen e do útero, o ruído gerado pela movimentação do eletrodo devido à respiração e aos movimentos da paciente, e ainda a interferência advinda da rede elétrica e de outros equipamentos eletrônicos. Todos esses aspectos tornam a extração do FECG um desafio (CLEMENTE et al., 2011).

Se considerarmos somente o MECG, já que os outros ruídos são comuns na obtenção da maioria dos sinais biológicos, iremos notar, conforme mostra a Figura 5, que ele muitas vezes sobrepõe o FECG, fazendo com que uma filtragem simples não seja suficiente para separar o sinal. Então, métodos mais complexos de processamento são necessários para extrair o FECG a partir do AECG (RASU; SUNDARAM; SANTHIYAKUMARI, 2015).

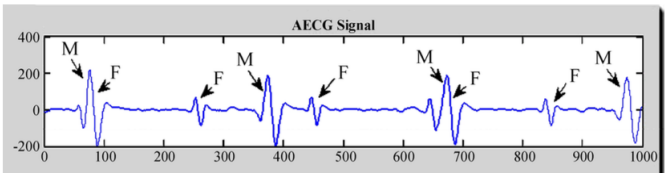

Figura 5 – Forma de onda do AECG Fonte:(HASAN; IBRAHIMY; REAZ, 2009)

2.2 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) surgem como uma forte proposta de solução para o problema apresentado. Podemos resumi-las como sendo aproximadores universais que podem aprender, generalizar, adaptar-se e gerar uma informação baseada na experiência e no contexto em que elas se encontram. Por meio dessas características, informações diversas podem ser estimadas. A depender da complexidade da tarefa, varia-se a arquitetura e o número de neurônios e camadas que ela possui para uma melhor aproximação (MRUGALSKI, 2013).

As RNAs são atualmente aplicadas em diversas áreas da tecnologia devido ao seu poder único de computação, que é capaz de reconhecer padrões, processar sinais, monitorar processos, realizar diagnósticos de falhas, controle de tolerância a falhas e aplicações em todas as áreas da engenharia. A disseminação e amadurecimento do conceito são confirmados pelo alto número de publicações nas últimas décadas envolvendo esse tema, provendo soluções para áreas de desenvolvimento tecnológico e para a indústria (MRUGALSKI, 2013).

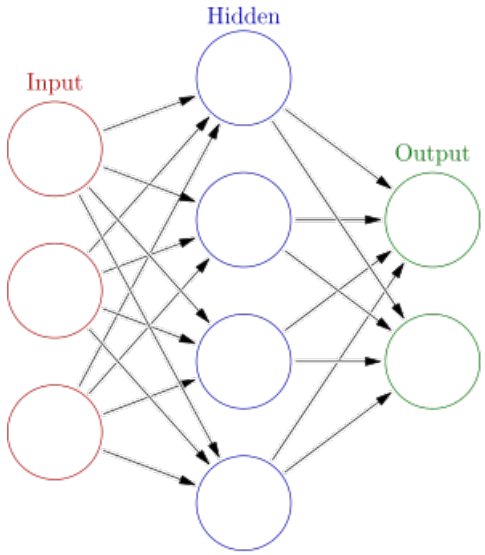

Figura 6 – Esquema de uma rede neural artificial. Fonte: (CASTROUNIS, 2017)

2.2.1 Definição

As RNAs foram pensadas e criadas para funcionarem como o cérebro humano. Uma RNA é formada por nós, ou neurônios, bem organizados e divididos em camadas que são, geralmente, sequencialmente conectadas, conforme mostra a Figura 6. Essas camadas são classificadas em camada de entrada, camadas intermediárias e camada de saída. A partir da comunicação entre elas, os dados inseridos são tratados pelos neurônios, pensados, até que uma resposta é gerada como saída do sistema (G; SAHEBI, 2011).

Um neurônio é, basicamente, constituído de quatro componentes: conexões e pesos, junção somadora, *threshold* e função de ativação (MACKAY, 2003). Os itens a seguir contêm uma breve descrição de cada um deles e a Figura 7 mostra o modelo de um neurônio artificial.

- Conexões e pesos (W): As conexões ligam os neurônios da rede, transferindo uma informação da saída para a entrada do neurônio seguinte. Para cada conexão, existe um peso associado (fator multiplicador).

- Junção somadora: Define a forma como os dados serão tratados dentro do neurônio, comumente essa função é um somador simples que realiza uma combinação linear das entradas da seguinte forma:

$$\sum_{i=m}^{n} x_i.W_i := x_m.W_m + x_{m+1}.W_{m+1} + \cdots + x_n.W_n \tag{2.1}$$

- *Threshold* (*bias*): Define um ponto de partida que aumenta ou diminui a influência do dado na ativação do neurônio. É um valor fixo que é somado ao resultado da junção somadora.

$$bias + \sum_{i=m}^{n} x_i.W_i, \tag{2.2}$$

- Função de ativação ($f(u)$): diz respeito a uma modularização da saída do neurônio utilizando funções conhecidas, que vão limitar a saída a uma determinada faixa e/ou introduzir não-linearidade ao modelo.

Podemos definir, matematicamente, a saída de um determinado neurônio k da seguinte forma:

$$y_k = f(u_k) = f(bias_k + \sum_{i=m}^{n} x_i.W_{ki} \tag{2.3}$$

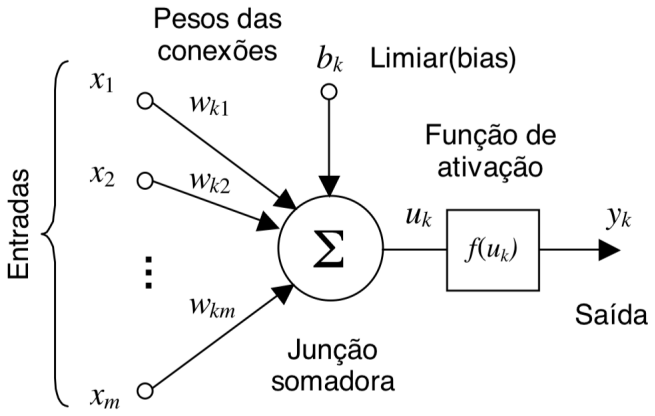

Figura 7 – Modelo de um Neurônio Artificial. Fonte: (CASTROUNIS, 2017)

2.2.2 Arquiteturas

Muitas são as possibilidades de se estruturar os neurônios em uma rede neural. O que deve ser levado em consideração primeiramente na escolha da arquitetura é o propósito para o qual a rede é designada. Conhecendo o propósito, podemos projetar a rede com o modelo que é mais adequado. Neste trabalho, descreveremos os 3 tipos de arquitetura de RNAs supervisionadas que são mais comumente utilizados e que abrangem a maioria das soluções.

2.2.2.1 Redes Neurais Artificiais Simples

RNAs simples são as redes que implementam as características mais básicas de uma rede neural, que foram descritas na seção 2.2.1. Elas necessariamente seguem um fluxo de dados na forma direta, da primeira camada para a segunda, e assim sucessivamente até a última camada, e todos os seus neurônios são interconectados de camada para camada. Além disso, essas redes podem ser constituídas de uma só camada, como é o caso da rede *Perceptron*, até várias camadas como o caso das redes *Multi-Layer Perceptron* e *Madaline*. São geralmente utilizadas para realizar aproximações, solucionar problemas estocásticos, e realizar classificações simples. A Figura 6 indica a organização de uma RNA Simples (ROSA, 2013).

Em uma RNA simples, a camada de entrada possui um neurônio para cada parâmetro, ou classe, que a base de dados possui. Esses neurônios são responsáveis por receber esses dados e enviá-los para a primeira camada intermediária. As camadas intermediárias, por sua vez, são responsáveis por processar esses dados e retirar características que possam ser distinguidas em cada amostra que for recebida. Cada camada intermediária pode ser constituída de vários neurônios, onde cada neurônio é conectado com todos os outros das camadas anterior e posterior, porém são independentes entre si. A camada de saída deve possuir o número de neurônios de acordo com as possibilidades de resultados esperados. Para o caso de uma classificação simples (binária), um único neurônio provê a

resposta da rede (LI; KARPATHY, 2015).

Uma das desvantagens das RNAs simples é o fato de que elas não escalonam muito bem quando a entrada é uma imagem. Por exemplo, uma imagem de 32x32x3 em uma rede comum com apenas um neurônio na camada intermediária precisaria de 3072 conexões em um único neurônio. Entretanto, como muitos outros neurônios são utilizados, esse valor cresce muito rapidamente. Claramente, nem todas essas conexões são necessárias e podem levar a um *overfitting*[1] da rede. Por essa razão, faz-se necessário o uso de redes convolucionais para solucionar esses casos. (LI; KARPATHY, 2015)

2.2.2.2 Redes Neurais Convolucionais (CNN)

As redes convolucionais, utilizadas em sua vasta maioria para processamento e classificação de imagens, são especialistas em identificar características das amostras que lhe são apresentadas, como podemos verificar na Figura 8. A sua organização é similar à das RNAs simples: são compostas de neurônios que possuem ligações com pesos ajustáveis e um valor inicial, onde cada neurônio recebe várias entradas e realiza uma operação matemática, geralmente, adicionando alguma não linearidade. Além disso, são redes de alimentação direta. A grande diferença entre a RNA simples e as RNC é que, essa última, assume que a entrada, necessariamente, é um vetor ou uma matriz com características lineares (imagens possuem linearidade entre os pixeis), o que faz com que o arranjo de pesos da arquitetura siga um padrão, reduzindo os parâmetros multiplicadores da rede e tornando o seu ajuste mais fácil (LI; KARPATHY, 2015).

As CNN possuem os seus neurônios dispostos em uma, duas ou três dimensões, a depender do tipo de dado com que ela está lidando. Além disso, a disposição das conexões nessas redes não são como as RNAs simples. Neste caso, cada neurônio está interessado somente em uma pequena parte da informação e se especializa naquele fragmento de cada amostra. Essas características permitem às CNN lidarem com um número de entradas muito elevado sem diminuir o desempenho, pois as conexões são reduzidas. Uma prática comum em CNN é fazer com que ocorra uma diminuição do número de amostras a cada camada que o dado atravessa, a fim de filtrar a informação que seja realmente significante para a classificação. (LECUN; KAVUKCUOGLU; FARABET, 2010)

As CNN são organizadas em camadas específicas que possuem uma função bem distinta entre si, diferentemente das RNAs simples, onde todas as camadas possuem operações muito similares. A Figura 9 mostra um modelo de CNN e a disposição das camadas, que são: camada de convolução, camada de não linearidade, camada de *pooling*, camada *flatten*, camada densa (*fully-connected*) e camada de saída. (LECUN; KAVUKCUOGLU; FARABET, 2010) Uma breve descrição da atuação de cada uma dessas camadas se en-

[1] Ocorre quando a rede se especializa em uma determinada característica dos dados, em prejuízo de outras características importantes.

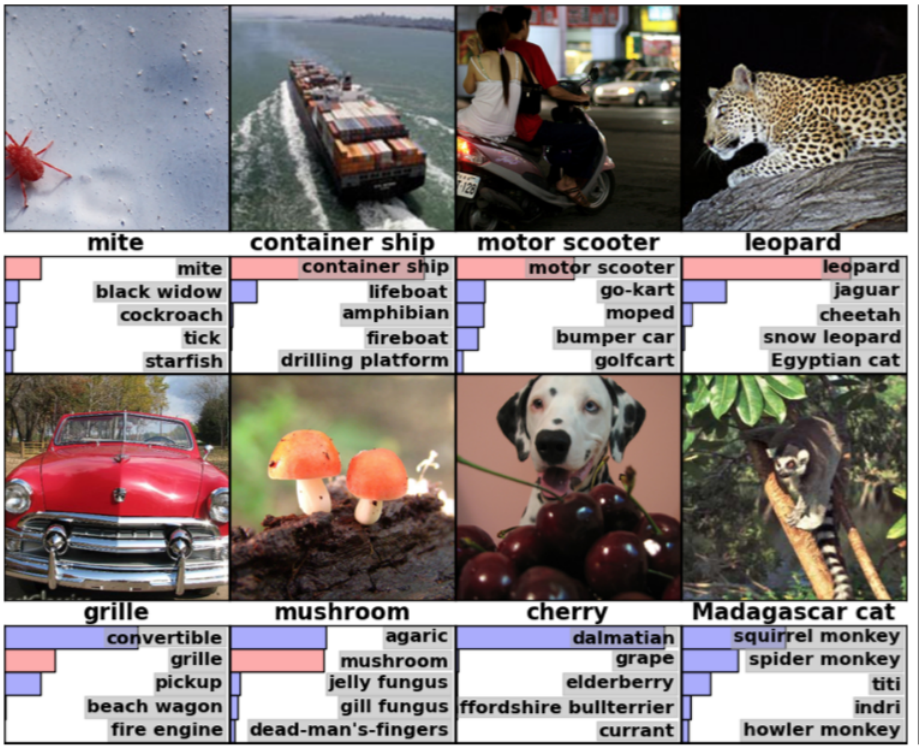

Figura 8 – Resultado da classificação de uma CNN. Fonte: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

contra a seguir.

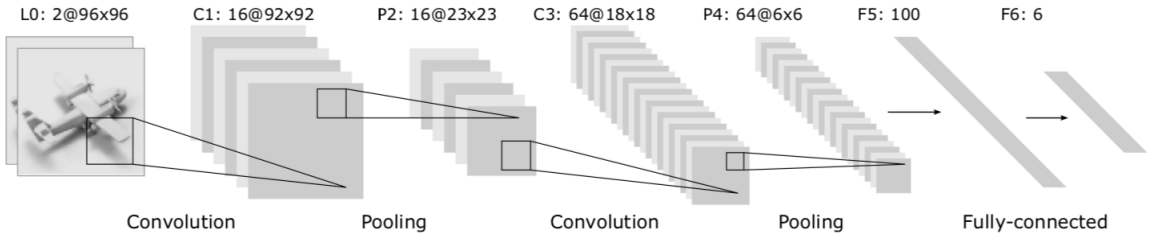

Figura 9 – Modelo de uma Rede Convolucional. Fonte: (SCHERER; MÜLLER; BEHNKE, 2010)

- Camada de convolução - É responsável por receber os dados. Cada neurônio trabalha com um único fragmento da amostra, aplicando um ou mais filtros para extrair alguma característica da mesma. Em seguida, repassa as novas amostras, já com o(s) filtro(s) aplicado(s), para a próxima camada.

- Camada de não linearidade[2] - É responsável por diminuir os efeitos adversos da linearidade em redes neurais, como no caso de imagens e séries temporais. Os neurônios dessa camada são chamados de (ReLUs) *Rectified Linear Units*.

[2] Alguns autores não a consideram como uma camada, mas sim como a função de ativação do neurônio da camada de convolução.

- Camada de *pooling* - Recebe as amostras da camada de não linearidade e toda a parte não-significativa da amostra é retirada, de forma que apenas o que está acentuado na amostra é preservado. Sendo assim, um fragmento menor é repassado para a próxima camada, porém apenas com dados de significância, acentuando as características encontradas na camada de convolução.

- Camada *flatten* - É responsável por reorganizar os dados e torná-los próprios para servirem de entrada para a camada densa, que por sua vez representa exatamente uma RNA simples. Como as camadas anteriores podem possuir mais de uma dimensão e ainda estarem organizadas em vetores e matrizes, nessa camada todos os dados são alinhados e concatenados.

- Camada densa - Trata-se de uma RNA simples, que vai receber os dados aprimorados nas camadas anteriores e, a partir deles, generalizar e identificar os nós realmente importantes para o resultado final.

- Camada de saída - É a camada que vai gerar o resultado final da rede, recebendo todo o processamento prévio e indicando aquilo que a rede identificou como resposta. A saída pode ser composta de um ou vários neurônios, a depender da quantidade de classificações que o projetista definiu, sendo que um neurônio é necessário para cada grupo.

2.2.2.3 Redes Neurais Recorrentes

As redes neurais recorrentes (RNNs) propagam informação tanto de forma direta quanto de forma inversa, permitindo que dados presentes nas camadas mais interiores influenciem diretamente as primeiras camadas. Em outras palavras, um neurônio de qualquer camada pode estar conectado com quaisquer outros neurônios, inclusive consigo mesmo. Esse processo de retroalimentação faz com que a adaptação da rede ocorra antes mesmo que uma saída seja gerada, aumentando o poder de aproximação desse tipo de rede. Em contrapartida, as RNNs são mais complexas e exigem muito mais poder computacional do que uma rede de alimentação direta (YI, 2013).

Dentre as várias arquiteturas possíveis de RNNs, a que tem tido o maior destaque recentemente, sendo usada inclusive por grandes companhias como a Google, Apple e Amazon para reconhecimento de padrões, é a rede *Long Short-Term Memory* (LSTM). Essa rede tem o poder de reter informação ao longo do tempo e utilizar essa informação posteriormente em outros contextos. Isso quer dizer que a rede é resistente a intervalos sem informações válidas e ainda que as informações armazenadas não sejam modificadas de forma iterativa, mas arbitrária, durante o processo de aprendizagem da rede. (OLAH, 2015)

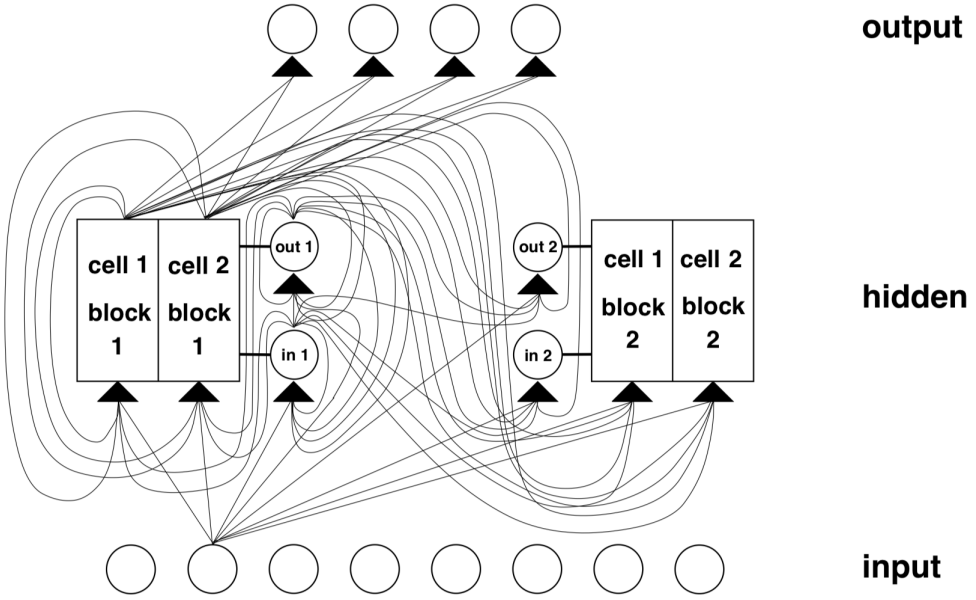

Figura 10 – Modelo de uma Rede Recorrente. Fonte: (HOCHREITER; SCHMIDHUBER, 1997)

2.2.2.4 Treinamento por *Backpropagation*

O treinamento diz respeito ao aprendizado da rede neural e está relacionado à adaptação, correção, otimização e representação da mesma. O algoritmo *backpropagation* é um dos mais utilizados para o treinamento de redes neurais por ter um conceito simples, ser computacionalmente eficiente e se adaptar a várias arquiteturas que as redes neurais possam assumir.(LECUN et al., 1998)

A forma como uma rede neural aprende determinada tarefa pode ser compreendida seguindo o exemplo mostrado na Figura 11.

Uma função:

$$M(Z^p, W), \tag{2.4}$$

Onde Z^p é a p-ésima amostra e W representa o conjunto de pesos, ou parâmetros ajustáveis, representa o resultado da rede neural.

A função de custo:

$$E^p = C(D^p, M(Z^p, W)), \tag{2.5}$$

Mede a distância entre D^p (O valor real) e o valor gerado pela rede neural, que é $M(Z^p, W)$.

O cálculo da média da função de custo é representado por:

$$E^{media}(W) = \sum \frac{E^p}{i}, \tag{2.6}$$

Onde i representa o número total de amostras.

Então, a forma como a rede busca se especializar para realizar determinada tarefa é ajustando o conjunto W, que é o único conjunto parametrizável, a fim de minimizar o valor de $E^{media}(W)$ a cada iteração (LECUN et al., 1998).

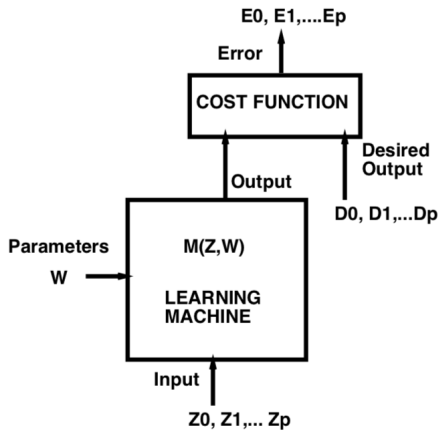

Figura 11 – Modelo de aprendizado de máquina Fonte: (LECUN et al., 1998)

O homem conhecido como o pai das redes neurais convolucionais, Yan LeCun, descreve o processo de treinamento de uma rede neural por meio do algoritmo de *backpropagation* da seguinte forma: "Projetar e treinar uma rede neural usando *backpropagation* exige que façamos muitas escolhas aparentemente arbitrárias como o número e tipo de nós, camadas, taxas de aprendizado, dados de treinamento e de testes, e assim por diante. Essas escolhas podem ser críticas, e no entanto não existe uma receita à prova de falhas para decidi-las, pois elas são altamente dependentes do problema e dos dados fornecidos"[3] (LECUN et al., 1998). Isso ilustra bem o quão empírico é o projeto de redes neurais.

2.3 Trabalhos Correlatos

Em 2009, HASAN; IBRAHIMY; REAZ, os autores tiveram por objetivo extrair o sinal FECG a partir do AECG utilizando métodos de processamento associados a um único neurônio. O princípio de funcionamento da solução proposta é o uso de filtros adaptativos através da combinação de uma rede adaptativa linear e um *Tapped Delay Line* (TDL).

[3] "Designing and training a network using backprop requires making many seemingly arbitrary chioces such as the number and types of nodes, layers, learning rates, training and test sets, and so forth. These choices can be critical, yet there is no foolproof recipe for deciding them because they are largely problem and data dependent."

De acordo com o funcionamento do TDL, o sinal de entrada, que é o sinal do ECG materno, é inicialmente armazenado em um vetor, em uma posição $n-1$. Quando o dado é requerido, o vetor passa o valor atual da entrada e o valor anterior, que estava posicionado em $n-1$. Esse vetor é consumido na rede adaptativa linear e logo depois é subtraído do sinal original AECG, o que resulta, em condições ideais, no FECG. Esse mesmo sinal serve de retroalimentação para atualizar os coeficientes internos da rede.

Já o trabalho conduzido por RASU; SUNDARAM; SANTHIYAKUMARI foi feito completamente no ambiente *LabView*, com sinais artificiais gerados no próprio *software*. O objetivo do trabalho foi obter a FHR usando um processamento de sinal adaptativo por meio do método dos mínimos quadrados. Com o intuito de eliminar a tensão da rede de 60Hz, foi utilizada uma técnica de baixo consumo de energia baseada em um filtro FIR de fase linear.

RASU; SUNDARAM; SANTHIYAKUMARI atesta que o seu trabalho usando filtros adaptativos supera os problemas advindos de uma amplitude limite, conseguindo detectar picos sobrepostos, como é o caso do FECG com o MECG. Aliado a isso, o autor mostra ter conseguido monitorar de maneira avançada um sinal de FECG. Tanto o processamento do sinal da mãe quanto o do sinal do bebê são realizados por filtros FIR baseados em um algoritmo adaptativo de LMS (*Least Minimum Square*), que é usado para estimar a FHR e calcular também o período do sinal. O FECG foi adquirido de um sinal AECG gerado usando sinais no *LabView*.

O trabalho de KARVOUNIS; TSIPOURAS; FOTIADIS obteve os melhores resultados entre os artigos pesquisados, atingindo 95,45% de acerto para sinais reais. A sua solução se baseia em uma metodologia de três estágios. Em cada estágio, uma forma diferente de processamento de sinais é aplicada. No primeiro estágio, é utilizado um filtro passa-banda e uma técnica de espaçamento de fases para isolar o sinal do ECG da mãe e poder eliminá-lo do AECG. No segundo estágio, o sinal é tratado para reduzir o ruído e amplificar os picos do complexo qRs do feto. Em um último estágio, uma técnica de histograma é aplicada para identificar os picos do FECG, que estão sobrepostos pelo MECG, sendo assim possível medir a FHR.

A Tabela 1 apresenta as características dos algoritmos e resultados alcançados pelos autores mencionados.

Além dessas pesquisas e projetos mencionados, podemos citar também os outros trabalhos de conclusão de curso que foram conduzidos na Faculdade do Gama que integram o contexto em que este presente trabalho está inserido. Com intuito de obter o sinal do AECG e disponibilizá-lo para um FPGA, um módulo de leitura foi desenvolvido por TUTIDA. A parte de comunicação dos dados gerados no FPGA e um dispositivo móvel foi realizada por RODRIGUES. E, por fim, um módulo de processamento de sinal, que também é o objetivo do presente trabalho, foi desenvolvido por BARBOSA usando

Tabela 1 – Comparação de acerto por método de extração da FHR. Adaptado de (HASAN; REAZ; IBRAHIMY, 2011)

Autor	Técnica de processamento	Tipo de sinal	Acerto (%)
PIERI et al.	Filtros casados	real	65
MOONEY et al.	Algoritmo adaptativo	real	85
HASAN; REAZ; IBRAHIMY	RNA & Correlação	real	93.75
KARVOUNIS; TSIPOURAS; FOTIADIS	Metodologia de três estágios (Filtro passa bandas + Redução de ruído + Técnica baseada em histogramas)	simulado	98.61
KARVOUNIS; TSIPOURAS; FOTIADIS	Metodologia de três estágios (Filtro passa bandas + Redução de ruído + Técnica baseada em histogramas)	real	95.45
RASU; SUNDARAM; SANTHIYAKUMARI	LMS + Filtros FIR	simulado	–

a abordagem de filtros adaptativos LMS, que obteve resultados significativos para sinais simulados, porém a resposta utilizando sinais reais não foi satisfatória.

3 Metodologia e Ferramentas

3.1 Metodologia

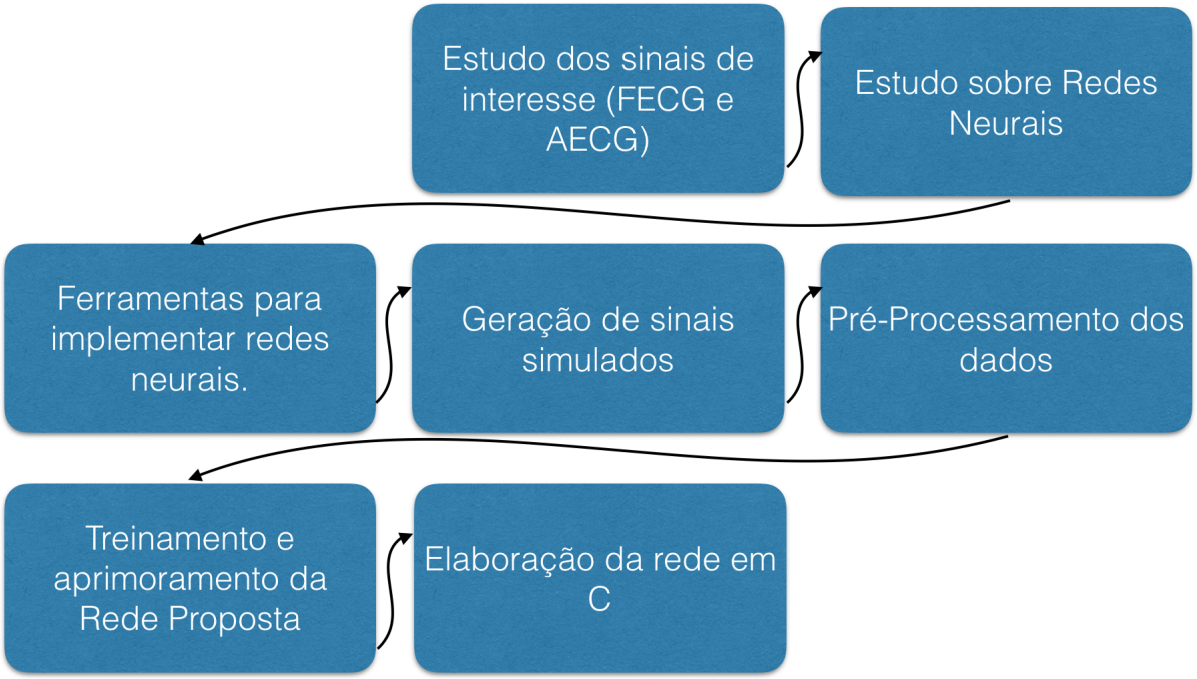

Figura 12 – Fluxo da metodologia utilizada no presente trabalho.

A execução desse trabalho, conforme a Figura 12 teve como ponto de início uma pesquisa bibliográfica que, no princípio, foi voltada a conhecer as peculiaridades do sinal de interesse, o FECG. Começando pelas características básicas de um eletrocardiograma e evoluindo para o desenvolvimento cardíaco fetal, foi possível compreender o comportamento do sinal, as diferentes formas de aquisição e as dificuldades que se inferem à detecção de picos do FECG, extraído do AECG.

Conhecendo o sinal de interesse e os desafios para processamento do mesmo, foi iniciada uma pesquisa para compreensão de como uma rede neural funciona e que diferentes arquiteturas seriam adequadas ao projeto. Após a familiarização com o conceito de redes neurais, mais fontes foram consultadas para serem usadas como base na proposta da arquitetura para este projeto.

Em seguida, foi feita a busca por ferramentas e formas de implementação de uma rede neural na solução de problemas simples. A tentativa de encontrar uma ferramenta eficiente para o projeto se iniciou com o *software* MATLAB, indo para o Spyder, utilizando-se de *backends* como Theano e TensorFlow, até chegar ao JupyterNotebook utilizando a API

[1] Keras. Já no quesito arquitetura, as primeiras tentativas foram a utilização de uma rede simples MLP (*Multi-Layer Percepton*. Depois para uma solução com redes recorrentes, entre elas redes LSTM. Daí foram implementadas arquiteturas de redes mistas (LSTM com redes convolucionais), até se chegar à arquitetura proposta como solução para esse projeto de uma rede convolucional.

Para testar as arquiteturas que foram desenvolvidas, foram gerados sinais genéricos, de maneira similar à realizada em BARBOSA, mas com adaptações. O *software* MATLAB foi usado para a criação de ondas que representassem o sinal do AECG, e ainda foi necessário que cada amostra gerada recebesse uma marcação, necessária para o treinamento da rede.

Então, considerando apenas a arquitetura que foi proposta como solução, foi realizado um pré-processamento dos dados para que se adequassem à ideia da solução. Nesta etapa, várias formas de treinamento e divisão de dados para treinamento e verificação foram testadas a fim de se encontrar a que apresentou o melhor resultado.

Como a rede foi gerada em Python, o passo seguinte foi a tradução para C, de forma a tornar possível sua execução em *hardware*. Foi necessário escrever manualmente todo o código executável, começando com a implementação de um modelo simplificado, que continha todos os tipos de camadas com dados conhecidos, a fim de validar o modelo. Posteriormente, o modelo foi utilizado para implementar a arquitetura completa.

3.2 Ferramentas

Neste item, as ferramentas utilizadas para a realização deste trabalho são brevemente descritas.

3.2.1 Matlab

O *software* **MatLab**, que é um acrônimo para *MATrix LABoratory*, possui uma linguagem de programação própria, voltada para estudos matemáticos e físicos. Essa linguagem é especial pelo falo de que a sua composição é baseada na forma como as equações e suas variáveis são escritas matematicamente. A sua alta performance e especifidade torna o **MatLab** um forte aliado para o cálculo e estudo nas áreas de processamento de sinais, cálculo numérico, álgebra linear e agora tem sido também largamente usado para projetos envolvendo redes neurais. (MATHWORKS, 2005)

O **MatLab** foi importante neste projeto tanto na fase de geração de dados simulados, quanto nos testes de possíveis arquiteturas.

[1] Interface de programação de aplicações formada por uma série de convenções e protocolos para desenvolvimento de *software*

3.2.2 *Jupyter Nootebook*

O projeto **Jupyter** foi criado para desenvolver *open-source software* e promover a possibilidade de computação interativa para dezenas de linguagens de programação. O **Jupyter Notebook** é uma aplicação também *open-source* desse projeto que pode ser utilizada tanto na *web*, quando nativamente em uma máquina. Essa aplicação é geralmente utilizada para desempenhar simulações numéricas, modelos estatísticos, visualização de dados, e para implementação de redes neurais. Além disso, ele é derivado do **Projeto IPython**, e por isso é um poderoso compilador para scripts em Python (KLUYVER et al., 2016).

Foi utilizado nesse projeto para simulação, implementação e visualização da rede neural proposta como solução, bem como de outras arquiteturas que não tiveram sucesso.

3.2.3 Keras

Keras diz respeito a uma API[2] para desenvolvimento de redes neurais utilizando Python. É capaz de rodar diferentes *backends* conhecidos no contexto de redes neurais como **Theano**, **TensorFlow**, **Caffee** e outros. Foi desenvolvido objetivando proporcionar rápida experimentação e pesquisa na área de redes neurais (KERAS. . . , 2018).

Essa API foi utilizada na implementação da rede neural proposta como solução desse projeto.

[2] Interface de programação de aplicações formada por uma série de convenções e protocolos para desenvolvimento de *software*

4 Arquitetura Proposta e Implementação da Rede Neural

Levando em consideração o que foi exposto na seção 2.2, entende-se que, por se tratar de um problema de detecção (ou classificação), as redes convolucionais são as mais indicadas. No entanto, pelo fato da informação que estamos classificando ser um sinal temporal e não ser comum encontrarmos redes convolucionais dedicadas a esse tipo de dado, as arquiteturas de redes comuns MLP foram primeiramente abordadas pelo fato de também realizarem classificação, tendo uma complexidade muitas vezes inferior aos outros tipo de rede. Arquiteturas de redes recorrentes foram testadas pelo seu excelente desempenho em séries temporais e por serem as redes com resultados mais promissores na atualidade para diversas aplicações. Arquiteturas que aliam diferentes tipos de rede também foram testadas. Por fim, mudar-se a abordagem do problema e adaptar os dados para uma forma possível de ser trabalhada em redes convolucionais, uma arquitetura possível para solução foi desenvolvida e é apresentada a seguir.

4.1 Arquitetura

As arquiteturas de redes neurais que são encontradas para predição e classificação de textos e de sinais de áudio são as que mais se assemelham à arquitetura proposta para aquisição da FHR no tocante ao tipo de entrada, pois também lidam com dados em uma e duas dimensões(DERIU et al., 2017). Além disso, essas redes possuem partes convolucionais e serviram de algum auxílio para propor uma rede neural que fosse capaz de detectar os picos de FHR. Esses exemplos foram considerados por não se encontrar uma arquitetura de igual propósito na literatura e, ao combinar características de uma arquitetura para dados 1D com arquiteturas para classificação de imagens, pode-se chegar à arquitetura aqui proposta.

A solução se baseia em uma rede convolucional de uma dimensão que pode ser aplicada para séries temporais e outros tipos de dados que apresentem características de continuidade, como é o caso do sinal de AECG. Essa rede possui 9 camadas e foi pensada para que, a partir unicamente do AECG, ela consiga identificar os picos e extrair a frequência cardíaca do bebê. A Figura 13 representa a solução proposta.

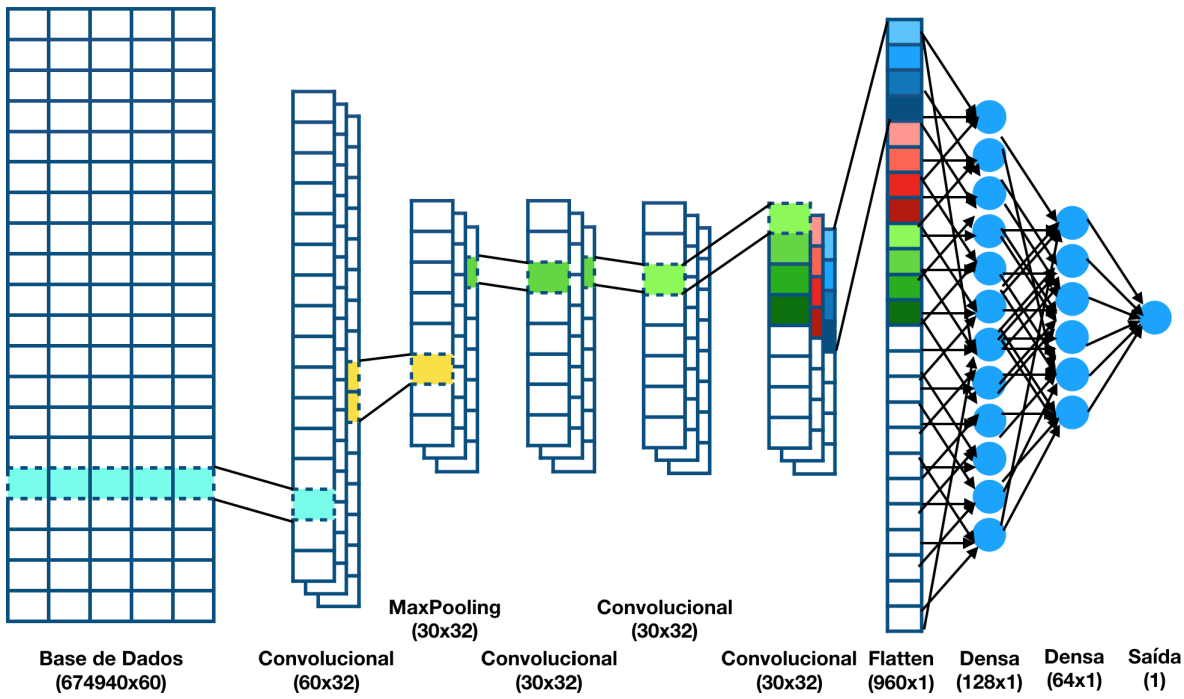

Figura 13 – Modelo da Arquitetura da Rede Convolucional proposta para a aquisição da FHR.

4.2 Implementação

Nesta seção, o fluxo completo para implementação da rede é descrito, bem como a função das camadas e dos filtros utilizados.

4.2.1 Onda de AECG gerada no Matlab

A fim de realizar a simulação da arquitetura, dados de AECG foram gerados utilizando o *software* **Matlab**, seguindo o trabalho de BARBOSA com as adequações necessárias para este projeto. Com o intuito de modularmos o sinal AECG, primeiramente obtemos o ECG materno. O *software* em questão contém uma função destinada à geração de ondas de ECG (**ecg()**), que recebe um parâmetro **L** que diz respeito ao comprimento desse sinal. O sinal que utilizamos, com $L = 675$ por exemplo, gera um sinal com 675 amostras que representa um ciclo completo de ECG. Além disso, a função **sgolayfilt()** é necessária para suavização da onda. A ação dessa função pode ser notada na Figura 14, que demonstra o sinal, antes e depois da suavização.

O mesmo processo foi feito para obtenção do sinal de ECG fetal (FECG). Porém, para o caso do bebê, o comprimento utilizado foi $L = 420$.

Para ambos os sinais mencionados (com $L = 675$ e $L = 420$), podemos extrair a frequência cardíaca correspondente de cada um. Para tal, primeiro devemos definir a taxa de amostragem do sinal que estamos interessados, que para o nosso projeto foi estabelecida

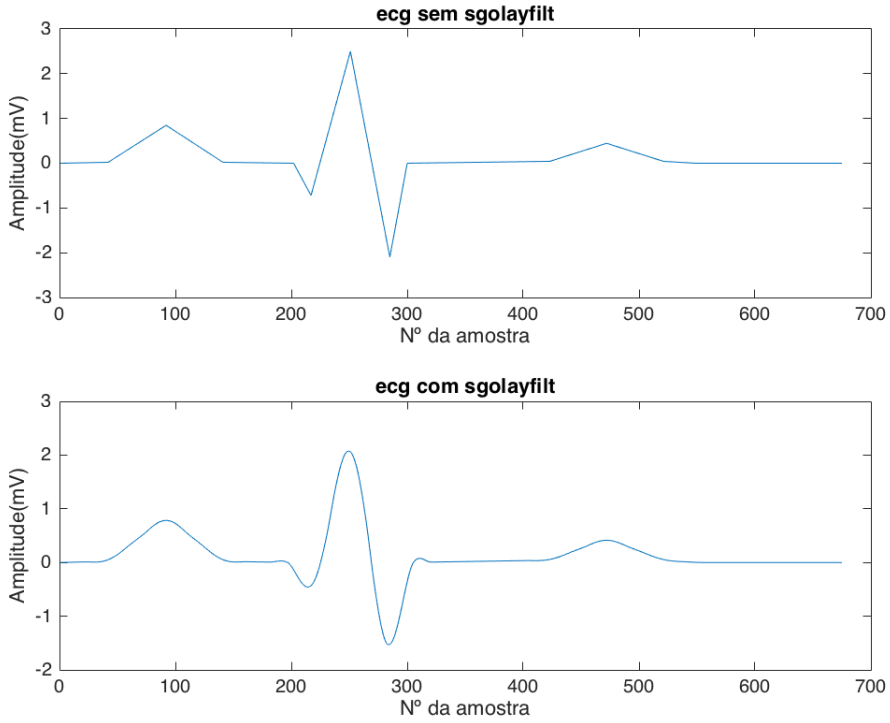

Figura 14 – Imagem superior representa a onda gerada usando a função ecg(675) e inferior a onda suavizada pela função sgolayfilt()

como 1KHz por segundo. A partir daí conseguimos extrair a frequência cardíaca segundo a expressão:

$$(Tx/n^oECG) * 60s = FC \tag{4.1}$$

Tx é a taxa de amostragem, **nºECG** é o número de amostras que um ciclo do sinal de ECG possui, e **FC** é a frequência cardíaca. Para o nosso projeto, temos os valores abaixo para o ECG materno e para o ECG fetal, respectivamente:

$$(1KHz/675) * 60 = 88.89BPM \tag{4.2}$$

$$(1KHz/420) * 60 = 142.86BPM \tag{4.3}$$

Esses valores conferem com os explicitados no capítulo 2, representando valores esperados durante a maior parte da gestação.

Além da diferença de frequência entre os sinais de ECG materno e fetal, um fator de 10 foi aplicado à magnitude dos sinais, de forma que o FECG é, aproximadamente, 11 vezes menor que MECG. O FECG tem o seu valor de pico em aproximadamente 0.181

mV, enquanto o MECG tem valor máximo em 2.075mV. A comparação da magnitude dos dois sinais pode ser vista na Figura 15

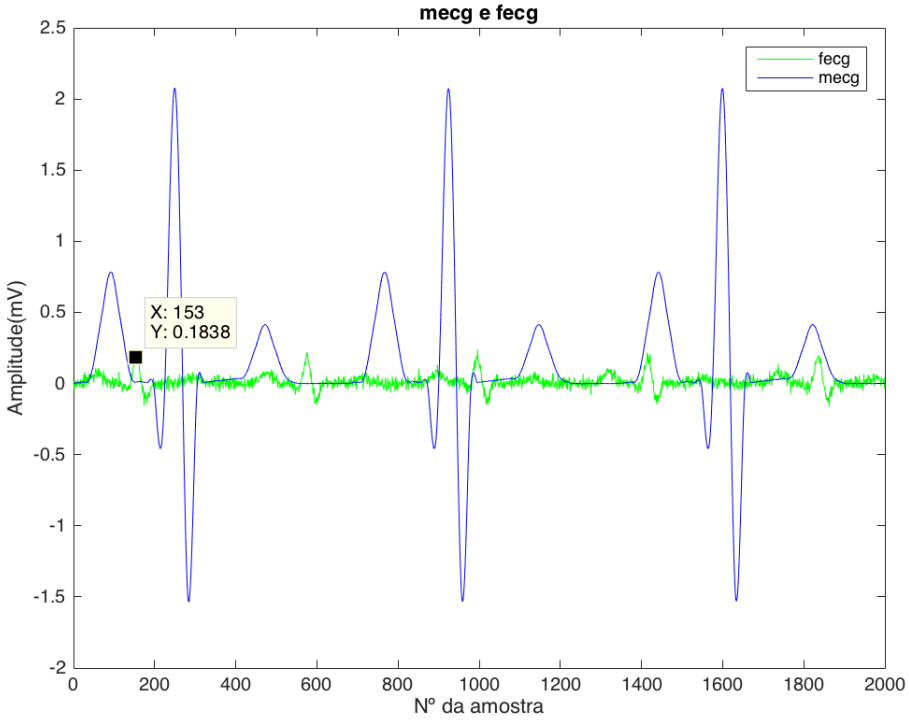

Figura 15 – Sinais FECG e MECG

Para atingir o objetivo principal, que é a geração de um sinal de AECG que representa um sinal real, ainda foram adicionados ruídos gaussianos brancos a cada um dos sinais de FECG e MECG. Após essa etapa, os sinais foram somados juntamente com um ruído aleatório. Então, o sinal AECG ficou definido da seguinte forma:

$$AECG = awgn(FECG) + awgn(MECG) + Noise \tag{4.4}$$

A função **awgn()** representa o ruído gaussiano branco e **Noise** representa o ruído aleatório. A Figura 16 mostra a forma de onda do AECG obtida.

Para o treinamento da rede, seiscentas e setenta e cinco mil amostras foram geradas e exportadas para um arquivo ***.csv**.

4.2.2 Pré-processamento dos dados

4.2.2.1 Classificação

Com o propósito de realizar o treinamento supervisionado da rede neural para identificar os picos do FECG utilizando somente o AECG, os dados primeiramente tiveram que ser marcados. Isso quer dizer que todos os 675.000 dados foram inicialmente

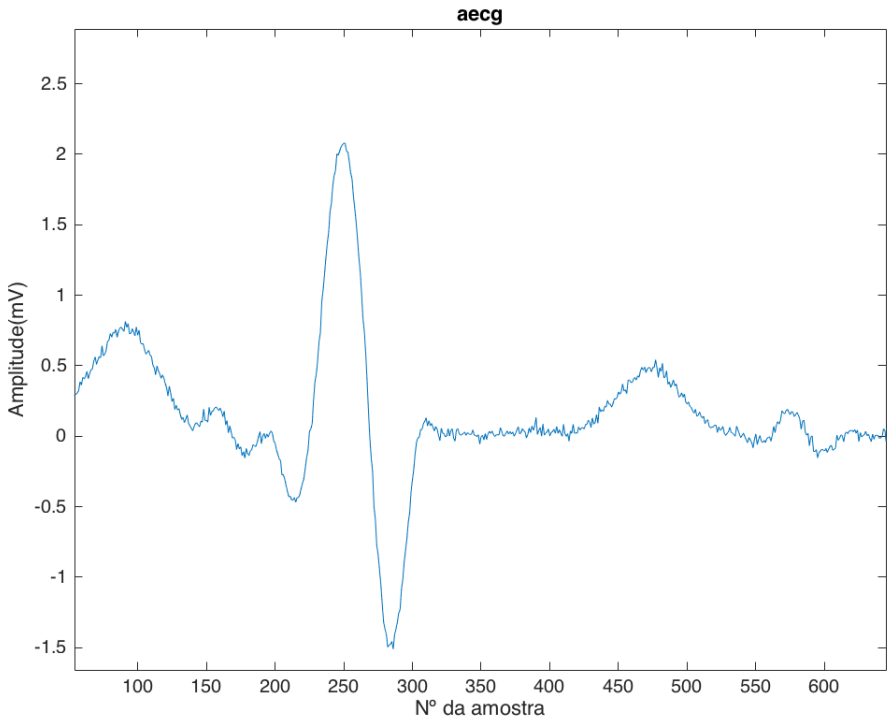

Figura 16 – Sinais AECG

classificados como sendo ou não um pico de FECG, onde 1 representa que aquela amostra é um pico de FECG e 0 representa que aquela amostra não é um pico. Isso foi só foi possível devido ao fato de termos gerado o FECG e, portanto, sabermos exatamente onde cada pico se encontraria, independente do ruído.

4.2.2.2 Formação dos conjuntos de amostras

Por estarmos utilizando uma arquitetura de uma rede convolucional, o esperado é que tratemos de um conjunto de amostras, geralmente imagens. Como os dados que geramos constituem uma sequência temporal, as amostras são todas singulares e, portanto, não podem ser classificadas por uma rede convolucional. O que foi proposto neste projeto foi o agrupamento das amostras de 60 em 60, de forma que o primeiro conjunto corresponda às 60 primeiras amostras, indo da amostra zero até a amostra 59, o segundo conjunto da amostra 1 até a 60, o terceiro da amostra 2 até a 61 e assim por diante, até a última amostra que seria da amostra 674941 até a amostra 675000, totalizando assim 674940 conjuntos contendo 60 amostras cada um.

Ainda, a fim de identificarmos se esses conjuntos contêm ou não um pico de ECG, eles foram todas classificadas mediante a sua 20ª amostra. Essa escolha se deve ao fato de ter proporcionado o melhor desempenho da rede em comparação a outras posições durante o treinamento. Dessa forma a rede considera não apenas um único valor, mas

todos os 20 valores anteriores e os 39 que estão depois na classificação dos conjuntos. A Figura 17 demonstra um conjunto em que a amostra 20 corresponde a um pico de FECG.

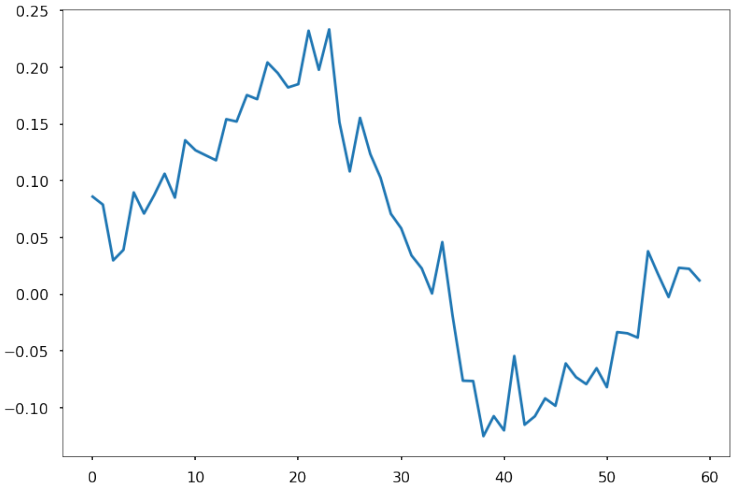

Figura 17 – Amostra classificada como tendo um pico de FECG

4.2.2.3 Balanceamento, embaralhamento e divisão dos conjuntos para treinamento e testes

Considerando que temos 675000 amostras, então, para um sinal de FECG com 420 amostras, temos 1607 batimentos fetais nesse sinal. Para que a rede receba um número balanceado de amostras positivas e negativas e não corramos o risco de que ela se especialize em identificar apenas as amostras negativas, que representam 99,76% das amostras, foi preciso realizar um tratamento desses dados. A forma que melhor se adequou à solução foi usar uma proporção de um terço de amostras positivas (1607) e dois terços de amostras negativas (3214).

Dos conjuntos resultantes do balanceamento, 60%(2892) foram destinados ao treinamento da rede e 40%(1929) para que a rede fosse testada. Essa proporção contribui para evitar o *overfitting* da rede. Ainda a fim de evitar o *overfitting*, os dados devem ser apresentados à rede de uma forma aleatória. As linearidades são um grande problema para as redes neurais, conforme explica a sessão 2.2. Então, por meio de um *script* disponível no Keras, os dados foram randomicamente embaralhados.

4.2.3 Rede Neural Proposta

Com as amostras do sinal de AECG classificadas, agrupadas, embaralhadas e divididas, temos o que precisamos para treinar a rede neural. Uma visualização da arquitetura da rede proposta foi apresentada na sessão 4.1, Figura 13. Outra representação está na Figura 18 que mostra o tipo da camada, o formato da saída de cada camada, e o número de parâmetros (pesos e bias) por ela utilizado.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 60, 32)            256
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 30, 32)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 30, 32)            5152
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 30, 32)            3104
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 30, 32)            1056
_________________________________________________________________
flatten_1 (Flatten)          (None, 960)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               123008
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 140,897
Trainable params: 140,897
Non-trainable params: 0
_________________________________________________________________
```

Figura 18 – Relatório do Keras informando a organização da rede neural

A seguir, temos um exemplo de como a rede trata um determinado conjunto de amostras, representado na Figura 19, e também a descrição da forma como cada camada atua e como elas foram organizadas para criar a rede neural.

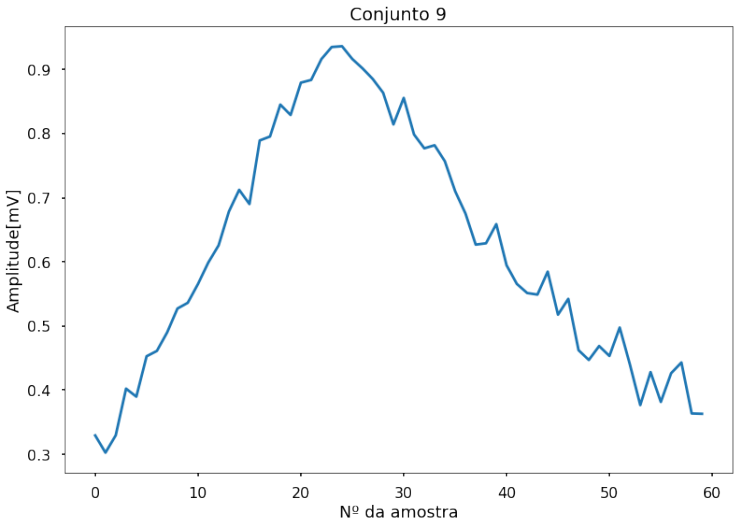

Figura 19 – Exemplo de conjunto de 60 amostras

- **1ª Camada, Convolucional:** Esta camada é responsável por receber os conjuntos de amostras, servindo de camada de entrada. Nela, são aplicados 32 filtros[1] de 7x1 aos conjuntos buscando, desde já, extrair características que sejam significativas na amostra. Esses filtros replicam as amostras 32 vezes, mantendo o tamanho

[1] conjuntos ou vetores de pesos

das mesmas, porém não a forma, e as encaminham para próxima camada. Como podemos notar pela Figura 20, a saída dessa camada possui a mesma quantidade de 60 amostras, porém agora com 32 formas diferentes.

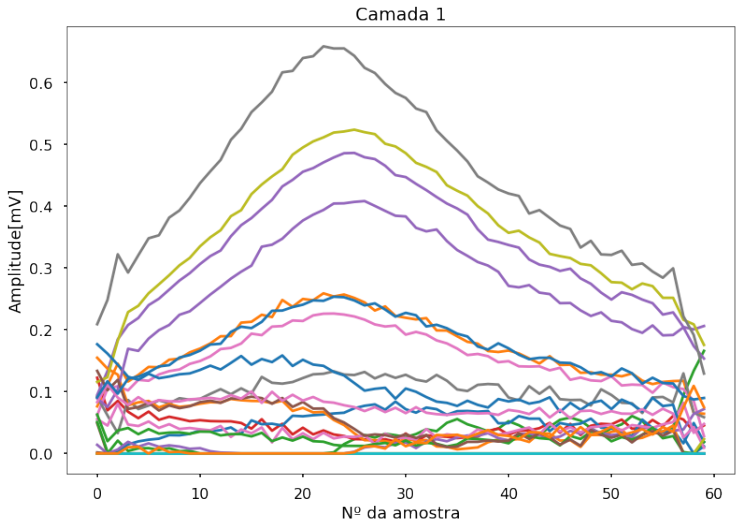

Figura 20 – Saída da camada 1

- **2ª Camada, MaxPooling:** Esta camada recebe os dados da camada anterior e aplica um novo filtro capaz de realizar o agrupamento das amostras de 2 a 2. A função dessa camada é realçar a característica que foi extraída na camada anterior, e ela faz isso retirando de 2 a 2 a amostra de menor importância (valor). Portanto, ela passa para a próxima camada a metade das amostras que recebeu. A Figura 21 mostra como as amostras agora caíram pela metade, de 60 para 30, mantendo porém as características principais do conjunto recebido.

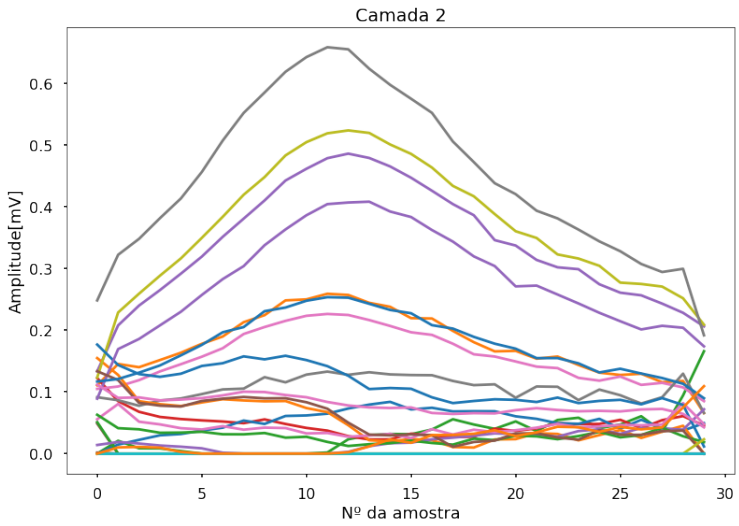

Figura 21 – Saída da camada 2

- **3ª Camada, Convolucional:** Esta camada tem a mesma atuação da primeira camada. São aplicados 32 filtros em cada um dos 32 conjuntos que ela recebe, porém agora os filtros são 5x1, buscando lapidar as formas de onda para extrair uma característica de interesse. A Figura 22 mostra a saída da camada e como a magnitude dos sinais aumenta a cada camada que passam, devido aos pesos (filtros) e bias que são aplicados.

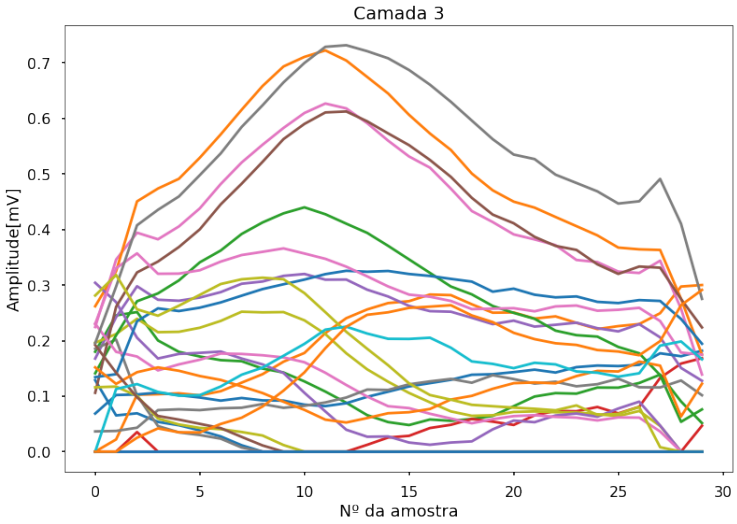

Figura 22 – Saída da camada 3

- **4ª Camada, Convolucional:** Esta camada tem a mesma atuação da camada anterior. São aplicados, novamente, 32 filtros nos dados que ela recebe, porém agora os filtros são 3x1, que conforme a experiência que adquiriram buscam refinar ainda mais as amostras recebidas. A Figura 23 demonstra a atuação do filtro sobre as amostras e como elas estão sendo repassadas para a camada seguinte.

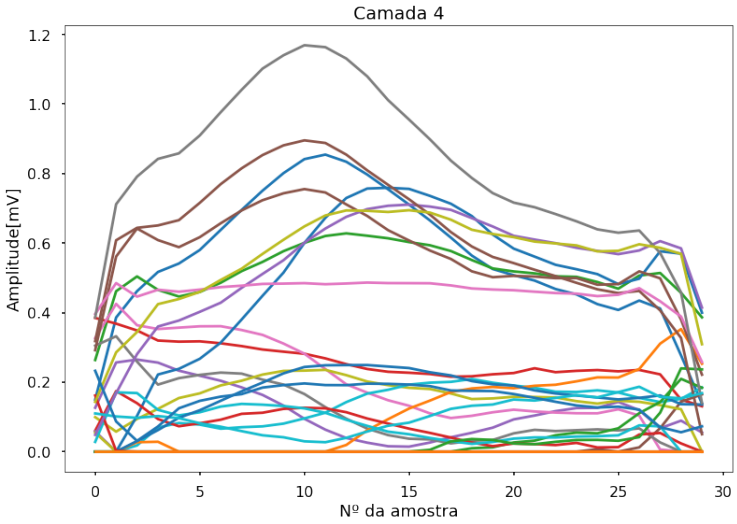

Figura 23 – Saída da camada 4

- **5ª Camada, Convolucional:** Atua da mesma forma que a camada anterior. 32 filtros são aplicados a cada conjunto de dados que ela recebe da quarta camada, porém agora os filtros são unitários, buscando extrair a característica que seja mais marcante na amostra, e assim atribuindo um valor a essa característica. A Figura 24 mostra como esse filtro modifica a amostra, como também a forma com que esses dados vão para a próxima camada.

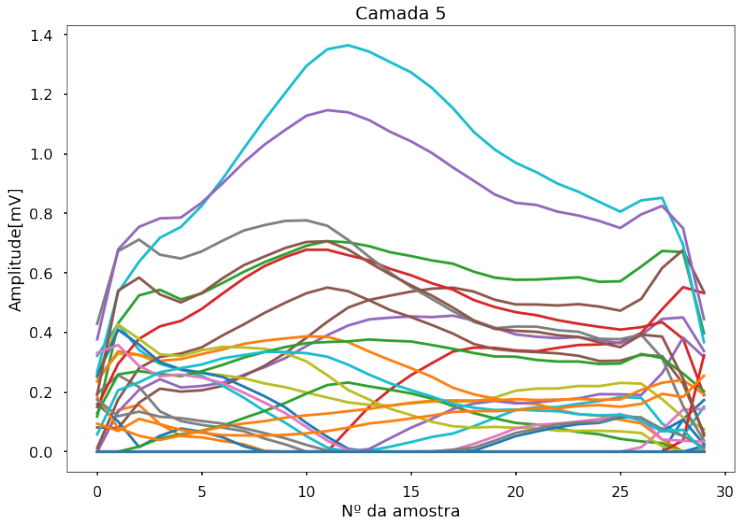

Figura 24 – Saída da camada 5

- **6ª Camada, *Flatten*:** Esta camada, como o seu nome sugere (*flatten* quer dizer achatar), alinha todos os dados recebidos da última camada convolucional concatenando-os, de forma que esses dados possam ser enviados para uma rede densa comum. A Figura 25 demonstra os valores de todas as 960 amostras (32 formas diferentes do conjunto original, multiplicado pelas 30 amostras de cada novo conjunto) que vem diretamente da quinta camada.

- **7ª Camada, Densa:** A partir daqui temos uma rede neural simples que vai atuar atribuindo importância para cada uma das características extraídas pela rede convolucional por meio das suas conexões e pesos. Essa camada é completamente conectada com a posterior. A Figura 26 mostra o valor que assume cada um dos 128 neurônios que compõem essa camada, e mostram a importância que cada neurônio tem para o resultado final.

- **8ª Camada, Densa:** Continua o trabalho da camada anterior, sendo mais específica em relação ao que a primeira densa fez, verificando quais os nós da camada anterior tem importância para o resultado final. A Figura 27 mostra os valores que assumem cada um dos 64 neurônios que compõem essa camada.

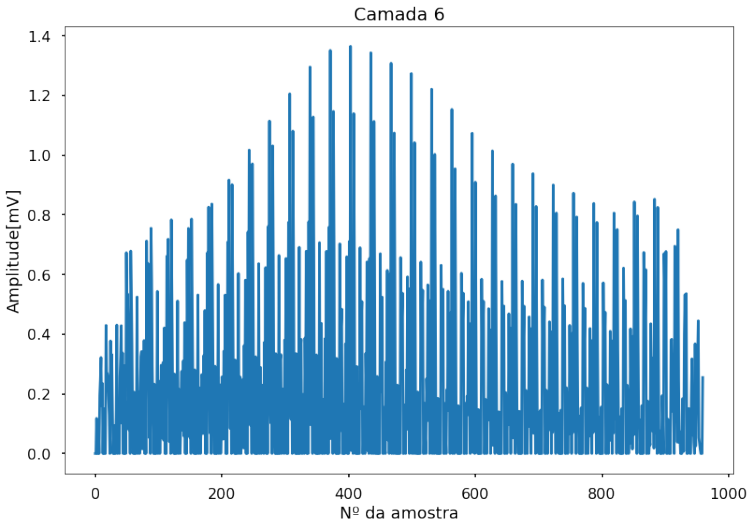

Figura 25 – Saída da camada 6

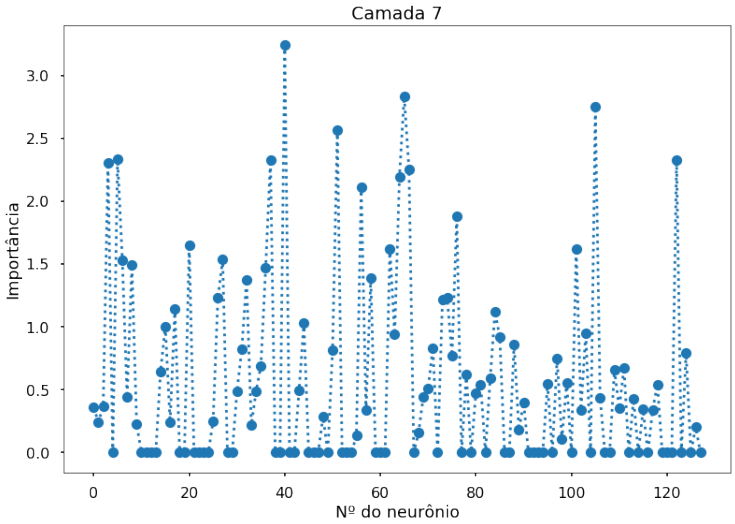

Figura 26 – Saída da camada 7

- **9ª Camada, Saída:** Enquanto todas as camadas anteriores têm como função de ativação ReLU[2], esta camada tem como função de ativação uma Sigmoid [3] para que agora, pelo produto interno dos nós da camada anterior, ela possa gerar um único valor entre 0 e 1 capaz de classificar o tipo de amostra que a rede recebeu. A Figura 28 exemplifica a classificação que o conjunto de exemplo recebeu. O valor que o neurônio de saída assume é de 93,38% de chances de existir um pico nesta amostra.

Após as 9 camadas da rede neural, o valor sugerido pela rede é comparado com o valor da classificação binária que o conjunto receberá previamente. Os algoritmos responsáveis pelo treinamento da rede, baseado nessas respostas geradas, estão descritos na

[2] *Rectifier Linear Unit*, representada pela função $f(x) = x^+ = \max(0, x)$

[3] $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$.

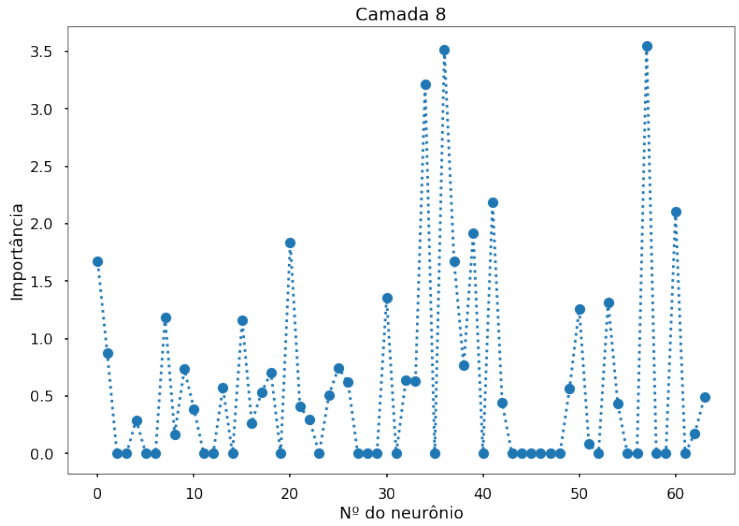

Figura 27 – Saída da camada 8

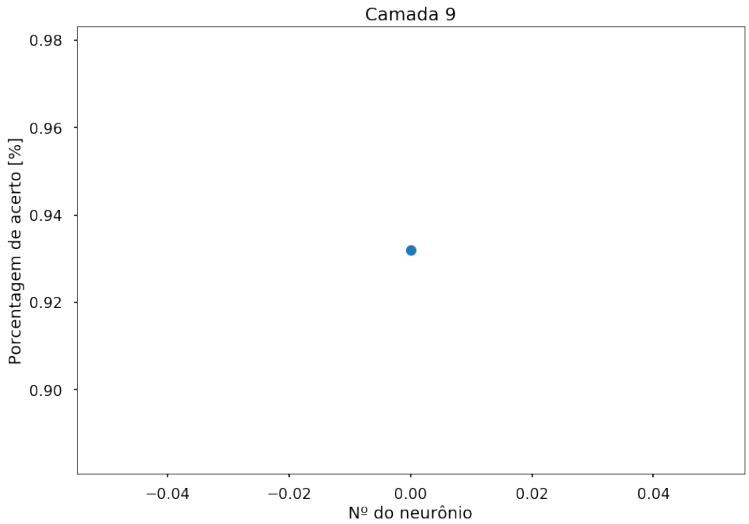

Figura 28 – Saída da camada 9

próxima seção.

4.2.4 Compilação e Treinamento da rede

O treinamento da rede foi feito com *batches*[4] de 128 agrupamentos de conjuntos, em 20 épocas.

O algoritmo de otimização utilizado, responsável por fazer os ajustes dos pesos e filtros da rede, foi o *Adam*, por ser o que obteve melhor desempenho em comparação aos demais. É geralmente usado em redes com um grande número de dados, e muito eficiente para otimização de problemas estocásticos.

[4] Número de amostras que são fornecidas à rede antes que ela ajuste os parâmetros dentro de uma época

A métrica utilizada no projeto para avaliar as respostas foi a *binary accuracy* que é comumente utilizada para projetos com uma resposta binária, pois ela compara o resultado gerado na rede com o resultado real binário fornecido pela anotação dos dados, arredondando o resultado para 1 ou 0.

Foi utilizada a função *binary crossentropy* para realizar a comparação entre as respostas da rede com a classificação prévia. Essa função busca medir a distância que o resultado apresentado pela rede está do resultado real e, baseada nisso, a rede rearranja os pesos das suas conexões para gerar um valor cada vez mais próximo. Os valores dessa função e das iterações do treinamento podem ser observados na figura 29

```
Epoch 1/20
2892/2892 [==============================] - 1s 312us/step - loss: 0.6181 - binary_accuracy: 0.6466
Epoch 2/20
2892/2892 [==============================] - 1s 296us/step - loss: 0.3666 - binary_accuracy: 0.8631
Epoch 3/20
2892/2892 [==============================] - 1s 295us/step - loss: 0.1750 - binary_accuracy: 0.9381
Epoch 4/20
2892/2892 [==============================] - 1s 383us/step - loss: 0.1329 - binary_accuracy: 0.9461
Epoch 5/20
2892/2892 [==============================] - 1s 333us/step - loss: 0.1183 - binary_accuracy: 0.9554
Epoch 6/20
2892/2892 [==============================] - 1s 304us/step - loss: 0.1026 - binary_accuracy: 0.9647
Epoch 7/20
2892/2892 [==============================] - 1s 297us/step - loss: 0.0887 - binary_accuracy: 0.9661
Epoch 8/20
2892/2892 [==============================] - 1s 295us/step - loss: 0.0881 - binary_accuracy: 0.9696
Epoch 9/20
2892/2892 [==============================] - 1s 295us/step - loss: 0.1007 - binary_accuracy: 0.9630
Epoch 10/20
2892/2892 [==============================] - 1s 393us/step - loss: 0.0842 - binary_accuracy: 0.9675
Epoch 11/20
2892/2892 [==============================] - 1s 352us/step - loss: 0.0672 - binary_accuracy: 0.9806
Epoch 12/20
2892/2892 [==============================] - 1s 372us/step - loss: 0.0605 - binary_accuracy: 0.9799
Epoch 13/20
2892/2892 [==============================] - 1s 428us/step - loss: 0.0571 - binary_accuracy: 0.9786
Epoch 14/20
2892/2892 [==============================] - 1s 395us/step - loss: 0.0561 - binary_accuracy: 0.9837
Epoch 15/20
2892/2892 [==============================] - 1s 365us/step - loss: 0.0582 - binary_accuracy: 0.9837
Epoch 16/20
2892/2892 [==============================] - 1s 295us/step - loss: 0.0692 - binary_accuracy: 0.9775
Epoch 17/20
2892/2892 [==============================] - 1s 425us/step - loss: 0.0522 - binary_accuracy: 0.9837
Epoch 18/20
2892/2892 [==============================] - 1s 299us/step - loss: 0.0442 - binary_accuracy: 0.9869
Epoch 19/20
2892/2892 [==============================] - 1s 388us/step - loss: 0.0488 - binary_accuracy: 0.9865
Epoch 20/20
2892/2892 [==============================] - 1s 436us/step - loss: 0.0582 - binary_accuracy: 0.9793
1929/1929 [==============================] - 0s 154us/step
Accuracy: 97.51%
```

Figura 29 – Resultado das iterações de treinamento

4.3 Implementação da rede em C

Toda a arquitetura apresentada foi modelada e validada utilizando a API Keras, no *software* Jupyter Notebook, com algum auxílio do *Spyder* para manipulação dos dados. Para efeito de desenvolvimento do código em C, uma versão simplificada da arquitetura

foi primeiramente implementada. Após o sucesso da implementação, foi desenvolvido o código em C correspondente à arquitetura proposta como solução.

Cada camada da rede é, basicamente, composta de multiplicações e somas sucessivas, como já foi mencionado anteriormente. Usando o procedimento disponível no Keras, $.layers[x].get\_weigths[y]$, podemos acessar os fatores multiplicadores da camada x, onde y igual a 0 retorna os pesos de cada um dos filtros, e para y igual a 1 obtemos o bias de cada camada. Esses fatores multiplicadores são aplicados, cada um conforme o neurônio e o filtro que estão associados, de forma semelhante ao bias. Um exemplo da implementação de uma das camadas convolucionais pode ser visto na Figura 30

```
115  void conv2()
116  {
117      // l - the number of samples
118      for(int l=0; l<32; l++)
119      {
120          //i - size of the samples
121          for(int i=0; i<sampleLength/poolLength; i++)
122          {
123              //j - number of filters
124              for(int j=0; j<numberOfFilters; j++)
125              {
126                  //k - size of the filters
127                  for(int k=0; k<secondFilterLength; k++)
128                  {
129                      secondConvOutput[l][i] += (secondConvInput[l][i+k]*secondConvFilter[l][j][k]);
130                  }
131                  // printf("%d\t", secondLayerOutput[j][i]);
132              }
133              secondConvOutput[l][i] += secondConvBias[l];
134              secondConvOutput[l][i] = RELU(secondConvOutput[l][i]);
135              // printf("\n");
136          }
137          // printf("\n");
138      }
139  }
```

Figura 30 – Implementação em C da segunda camada convolucional

5 Resultados

5.1 Resultados

Alguns dos resultados de mais relevância das classificações geradas pela rede são apresentados a seguir. Para todas as 1929 amostras, foi gerado um gráfico semelhante aos apresentados a seguir.

Para compreensão dos diagnósticos dos conjuntos apresentados, é necessário o conhecimento prévio das características do sinal FECG, MECG e AECG, principalmente em relação à amplitude de cada sinal, como também a frequência, mencionados na seção 4.2.1. Deve-se levar também em consideração que a classificação real dos conjuntos se dá sempre pela amostra de número 20.

Os valores de "Classificação real" de cada imagem correspondem à anotação mencionada na seção 4.2.2.1, que foi previamente feita em cada conjunto. Foram adotados como acerto valores de "Classificação da rede" maiores que 0.5, ou seja, 50%.

A figura 31, demonstra um pico de FECG visível. O sinal apresenta uma onda com máximo em aproximadamente 0.23mV, o que só pode ocorrer em um pico de FECG. A rede acertou, indicando 98,75% de chances de ser um pico de FECG.

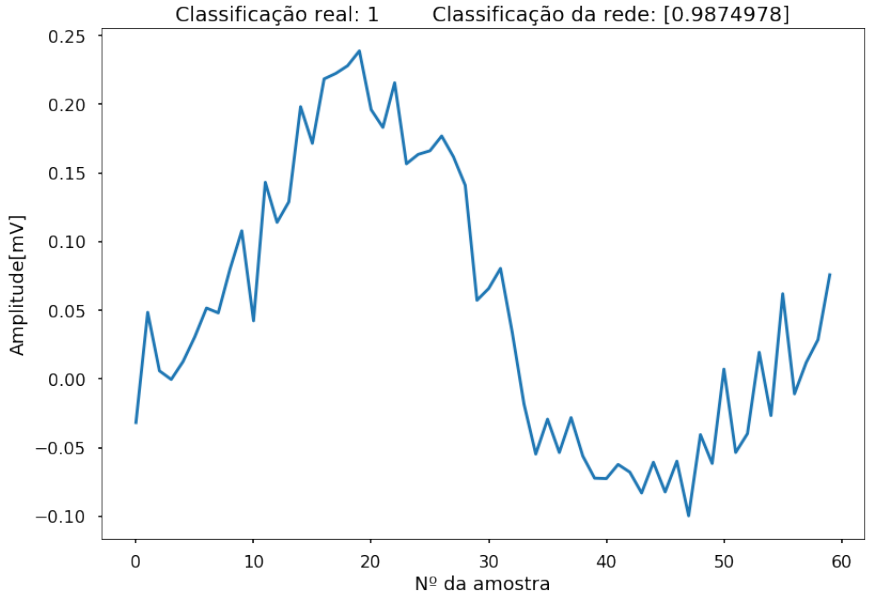

Figura 31 – Conjunto 1 de amostras

A Figura 32, demonstra, também, um pico de FECG visível em uma região perto de uma onda P. O sinal apresenta onda com valor máximo em aproximadamente 0.21mV,

evidenciando um pico de FECG. A rede indicou 99,65% de chances da amostra ser igual a 1, ou seja, um acerto.

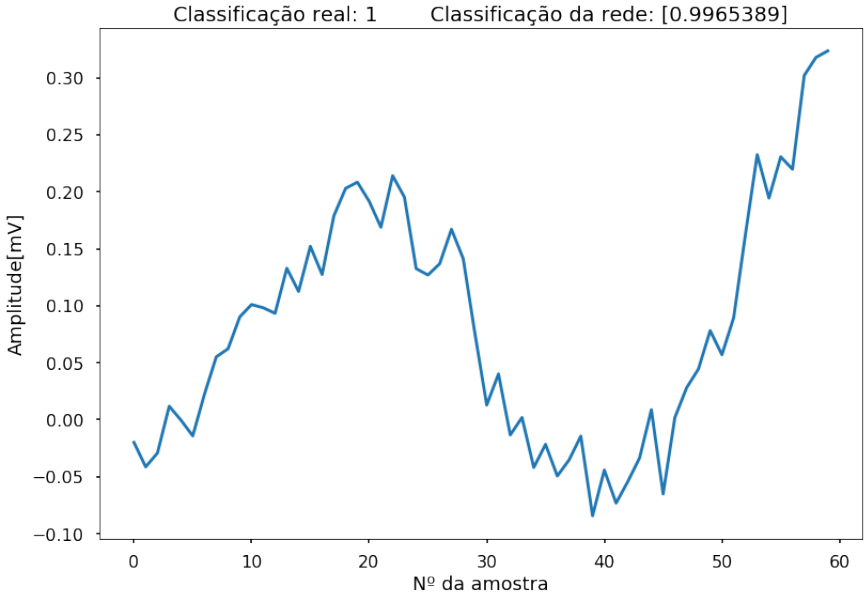

Figura 32 – Conjunto 2 de amostras

A Figura33, nos mostra um pico de FECG em meio à onda T. O sinal apresenta onda com máximo em aproximadamente.0.55mV. Considerando que o pico da onda T tem, em média, 0.4mV, acrescendo o do FECG que tem 0.18mV, temos aproximadamente os 0.55mV que está evidenciado. A rede acertou, porém com um baixo valor em porcentagem, resultando em 59,01% de chances de ser um pico de FECG.

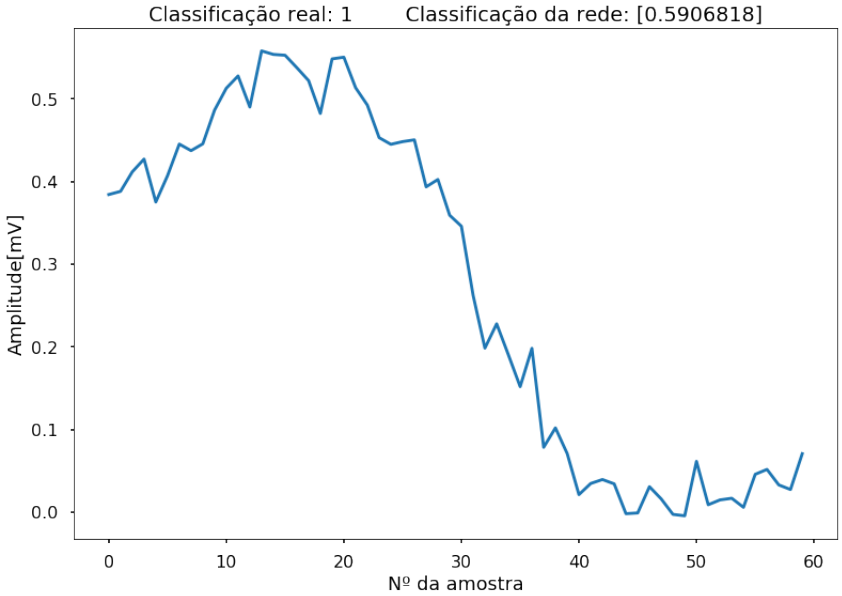

Figura 33 – Conjunto 3 de amostras

A Figura 34, representa um pico de FECG não visível. O sinal apresenta onda com

valor de máximo em aproximadamente 0.95mV, mostrando que o pico de FECG coincidiu com o cume da onda P, que assume valores até 0.8, sem considerar o ruído. A rede acertou, indicando 93,20% de chances de ser um pico de FECG.

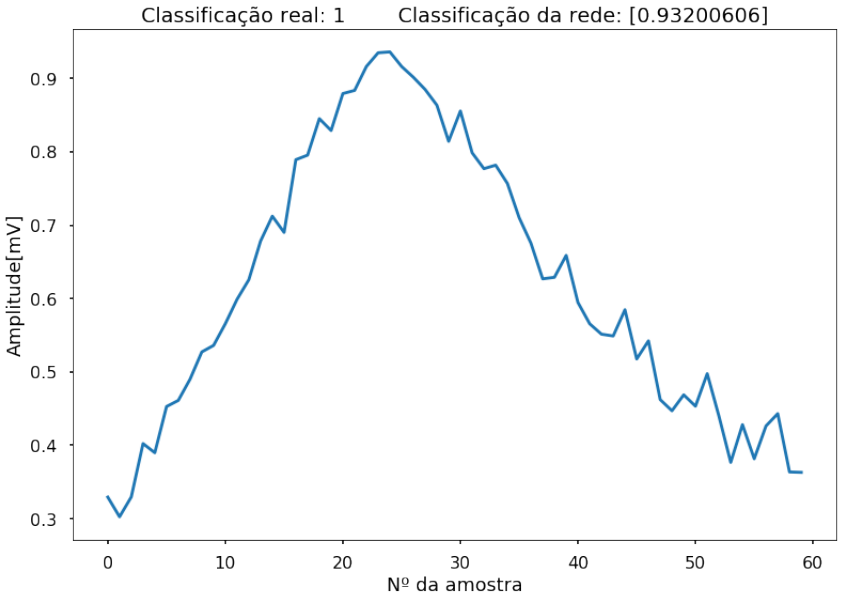

Figura 34 – Conjunto 4 de amostras

A Figura 35 retrata, também, um pico de FECG não visível. Nesse caso, o pico está no início do complexo QRS e não podemos notá-lo por estar coincidindo com o primeiro vale do complexo QRS. A rede acertou, resultando em 92,49% de chances de ser um pico de FECG.

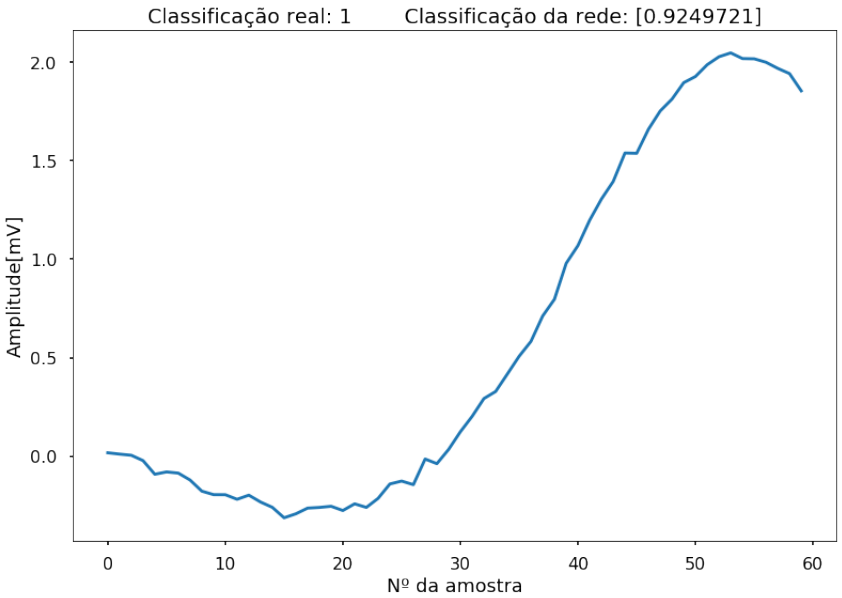

Figura 35 – Conjunto 5 de amostras

A Figura 36, demonstra um pico de ECG não visível. Nesta parte do sinal, a

magnitude da onda QRS, aliada ao ruído, encobre o sinal de FECG. A rede acertou, indicando 98,75% de chances de ser um pico de FECG.

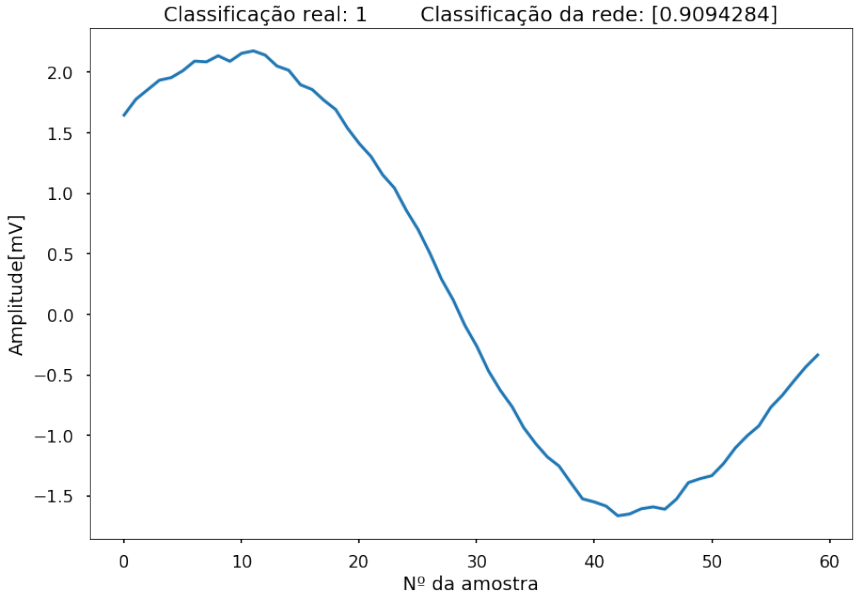

Figura 36 – Conjunto 6 de amostras

A Figura 37, diz respeito a um pico de FECG que não foi detectado pela rede, que classificou a amostra com 38,48% de chances de conter um pico de FECG. Porém, a amostra contém o pico, e o resultado esperado seria um valor acima dos 50%. Logo, a rede errou.

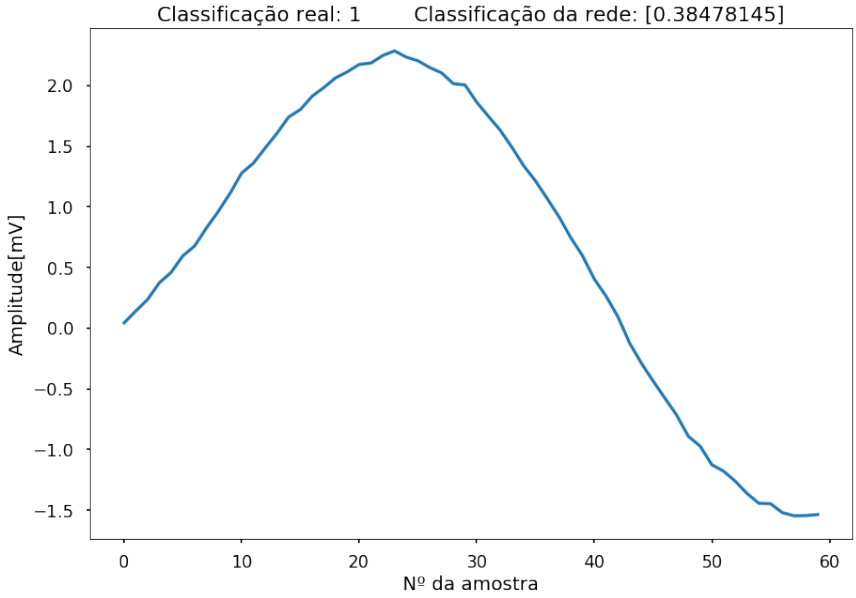

Figura 37 – Conjunto 7 de amostras

Na Figura 38, não existe um pico de FECG no conjunto. Ainda assim, a rede classificou a amostra com 59,26% de chances de conter o pico, ou seja, a rede novamente

errou a classificação.

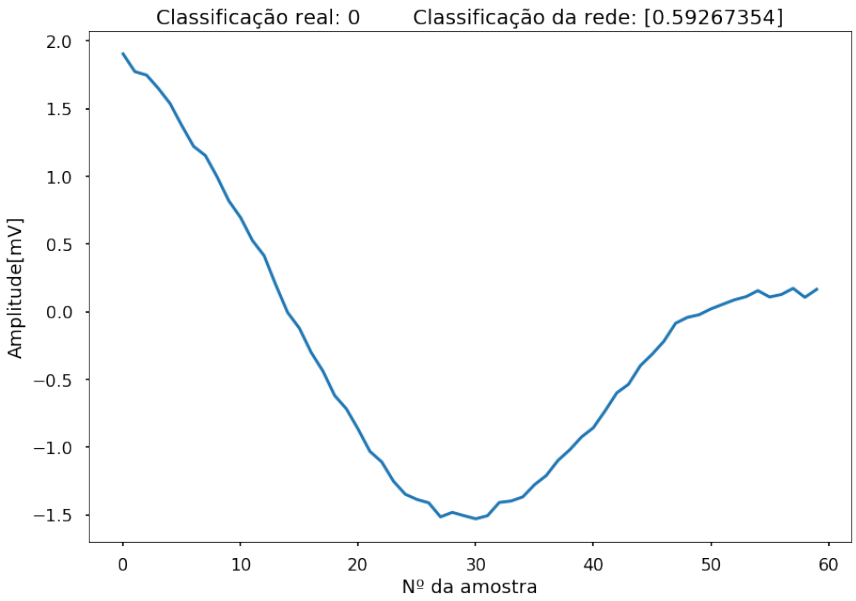

Figura 38 – Conjunto 8 de amostras

Utilizando os sinais simulados, a rede apresentou 96,78% como o menor percentual de acerto, e 98,36% como o maior, com um desvio padrão de 0,67% e valor médio igual a 97,43%. Considerando a média dos treinamentos realizados, para a verificação, tivemos, em média, 1288 amostras negativas[1], e desses a rede classifica, em média, 13 deles como sendo picos verdadeiros. Já para as amostras positivas[2], foram utilizadas, em média, 641 amostras, onde a rede acerta, em média, a classificação de 605 amostras. Esses números variam de treinamento para treinamento, de acordo com o embaralhamento dos dados previamente citado, mas a precisão se mantêm dentro do valor médio indicado.

A Tabela 2, apresenta uma comparação final, do presente trabalho com os demais citados. É preciso levar em consideração que a forma utilizada para gerar os dados simulados nos trabalhos comparados não são as mesmas.

Tabela 2 – Comparação entre o presente trabalho e os demais citados

Autor	Técnica de processamento	Tipo de sinal	Acerto (%)
PIERI et al.	Filtros casados	real	65
MOONEY et al.	Algoritmo adaptativo	real	85
HASAN; REAZ; IBRAHIMY	RNA & Correlação	real	93.75
KARVOUNIS; TSIPOURAS; FOTIADIS	Metodologia de três estágios (Filtro passa bandas + Redução de ruído + Técnica baseada em histogramas)	simulado	98.61
KARVOUNIS; TSIPOURAS; FOTIADIS	Metodologia de três estágios (Filtro passa bandas + Redução de ruído + Técnica baseada em histogramas)	real	95.45
RASU; SUNDARAM; SANTHIYAKUMARI	LMS + Filtros FIR	simulado	–
Presente Trabalho	**Rede Neural Convolucional**	**simulado**	**97.43**

[1] Amostras que não contém um pico de FECG, ou seja, que foram classificadas previamente com o marcador zero

[2] Amostras que realmente contém um pico de FECG

É preciso levar em consideração que a forma utilizada para gerar os dados simulados nos trabalhos comparados não são as mesmas.

Os dados reais que estão disponíveis hoje não atendem as necessidades do tipo de dado requerido para o devido treinamento da rede neural.

6 Conclusão

6.1 Conclusão

Este trabalho teve por objetivo estruturar uma rede neural capaz de detectar os picos do FECG em meio a um sinal de eletrocardiograma abdominal, e tornar possível mensurar dessa forma a frequência cardíaca fetal. Esse parâmetro, é fundamental para o monitoramento precoce do feto, e pode levar à diminuição dos índices de mortalidade fetal.

Até os dias de hoje, é um desafio a obtenção da FHR, devido à alta relação sinal-ruído do FECG. Várias abordagens têm sido estudadas, e implementadas, mas ainda não temos uma solução definitiva e comercial.

Esse trabalho consistiu em explorar um ramo relativamente novo em processamento de sinais, que, da forma como aqui se apresenta, uma rede neural convolucional, não foi encontrada em trabalhos similares.

As ferramentas utilizadas, em especial a API Keras e o Jupyter Notebook, se mostraram de grande utilidade na modelagem de redes neurais diversas, permitindo a simulação e prototipação de forma rápida e simplificada.

Redes Neurais Artificiais comuns como a rede *Multi-Layer Perceptron*, não apresentaram resultados significativos para detecção de picos de FECG, a partir do AECG. Ainda, Redes Recorrentes e Redes Mistas, dentre as arquiteturas experimentadas, não apresentaram resultados significativos.

A Rede Convolucional proposta atingiu excelentes resultados para dados simulados, 97.43% de acerto na classificação das amostras. Aliando o resultado obtido no presente trabalho, com os demais resultados e implementações reais que vemos em diversas áreas, temos o sentimento que resultados semelhantes podem ser obtidos para os dados reais. Porém, não foi possível realizar treinamento da rede com dados reais, visto que uma base de dados adequada não estava disponível.

Sugere-se que para próximos trabalhos, uma base de dados seja levantada com o auxílio de um segundo exame, que indique algum parâmetro rastreável que possa ser usado para guiar o treinamento da rede neural. Esses parâmetros podem ser tanto as curvas geradas pela cardiotocografia, que fornecem os momentos exatos do batimento cardíaco do bebê, quanto o momento dos batimentos que podem ser obtidos de maneira mais simplificada com um detector fetal. Sugere-se ainda, uma parceria com hospitais para o levantamento de uma base de dados adequada.

Referências

BARBOSA, I. J. T. Aceleração de algoritmos para estimativa da frequência cardíaca fetal utilizando fpga. 2016. Citado 3 vezes nas páginas 34, 38, and 42.

CARLSON, B. M. *Embryology and Developmental Biology*. 5. ed. Michigan: Saunders, 2013. Citado 2 vezes nas páginas 11 and 23.

CASTROUNIS, A. *Artificial Intelligence, Deep Learning, and Neural Networks, Explained*. 2017. Disponível em: <http://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html>. Citado 3 vezes nas páginas 11, 26, and 28.

CLEMENTE, R. et al. Fetal ecg extraction from maternal abdominal ecg using neural network. *IEEE Transaction on Bio-Medical Engineering*, v. 58, n. 2, 2011. Citado na página 25.

CREMER, M. *Über die direkte ableitung der aktionsströme des menschlichen herzens vom oesophagus und über das elektrokardogramm des fötus*. [S.l.]: Lehmann, 1906. Citado na página 19.

DAVIES, H. *Cardiotocography (CTG)*. 2012. Disponível em: <http://www.ebme.co.uk/articles/clinical-engineering/23-cardiotocography-ctg>. Citado 2 vezes nas páginas 11 and 24.

DERIU, J. et al. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 26th International Conference on World Wide Web*. [S.l.], 2017. p. 1045–1052. Citado na página 41.

FREEMAN, R. K. et al. *Fetal heart rate monitoring*. [S.l.]: Lippincott Williams & Wilkins, 2012. Citado 4 vezes nas páginas 11, 19, 20, and 24.

G, A. H.; SAHEBI, M. Combination of ga and ann to high accuracy of polarimetric sar data classification. *Advances in Computational Intelligence*, Springer, p. 207–214, 2011. Citado na página 27.

HASAN, M.; IBRAHIMY, M. I.; REAZ, M. B. I. Fetal ecg extraction from maternal abdominal ecg using neural network. *Journal of Software Engineering and Applications*, Scientific Research Publishing, v. 2, n. 05, p. 330, 2009. Citado 2 vezes nas páginas 25 and 33.

HASAN, M.; REAZ, M. B. I.; IBRAHIMY, M. I. Fetal electrocardiogram extraction and r-peak detection for fetal heart rate monitoring using artificial neural network and correlation. In: IEEE. *Neural Networks (IJCNN), The 2011 International Joint Conference on*. [S.l.], 2011. p. 15–20. Citado 3 vezes nas páginas 13, 35, and 59.

HASAN, M. A.; IBRAHIMY, M. I.; REAZ, M. B. I. An efficient method for fetal electrocardiogram extraction from the abdominal electrocardiogram signal. *Journal of Computer Science*, Science Publications, v. 5, n. 9, p. 619–623, 2009. Citado 3 vezes nas páginas 11, 25, and 26.

HATAI, I.; CHAKRABARTI, I.; BANERJEE, S. Fpga implementation of a fetal heart rate measuring system. In: IEEE. *Advances in Electrical Engineering (ICAEE), 2013 International Conference on*. [S.l.], 2013. p. 160–164. Citado na página 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado 2 vezes nas páginas 11 and 32.

HON, E. H. The electronic evaluation of fetal heart rate: Ii. changes with maternal hypotension. *American journal of obstetrics and gynecology*, Mosby, v. 79, n. 2, p. 209–215, 1960. Citado na página 19.

KARVOUNIS, E. C.; TSIPOURAS, M. G.; FOTIADIS, D. I. Detection of fetal heart rate through 3-d phase space analysis from multivariate abdominal recordings. *IEEE Transactions on Biomedical Engineering*, IEEE, v. 56, n. 5, p. 1394–1406, 2009. Citado 3 vezes nas páginas 34, 35, and 59.

KERAS Documentation. 2018. Disponível em: <https://keras.io>. Citado na página 39.

KLUYVER, T. et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In: *ELPUB*. [S.l.: s.n.], 2016. p. 87–90. Citado na página 39.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado 2 vezes nas páginas 11 and 30.

LECUN, Y. et al. Efficient backprop. In: *Neural networks: Tricks of the trade*. [S.l.]: Springer, 1998. p. 9–50. Citado 3 vezes nas páginas 11, 32, and 33.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: IEEE. *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. [S.l.], 2010. p. 253–256. Citado na página 29.

LI, F.-F.; KARPATHY, A. *Convolutional Neural Networks for Visual Recognition*. 2015. Citado na página 29.

MACKAY, D. J. *Information theory, inference and learning algorithms*. [S.l.]: Cambridge university press, 2003. Citado na página 27.

MATHWORKS, I. *MATLAB: the language of technical computing. Desktop tools and development environment, version 7*. [S.l.]: MathWorks, 2005. v. 9. Citado na página 38.

MOONEY, D. M. et al. Computer algorithm for adaptive extraction of fetal cardiac electrical signal. In: ACM. *Proceedings of the 1995 ACM symposium on Applied computing*. [S.l.], 1995. p. 113–117. Citado 2 vezes nas páginas 35 and 59.

MRUGALSKI, M. *Advanced neural network-based computational schemes for robust fault diagnosis*. [S.l.]: Springer, 2013. v. 510. Citado na página 26.

MURPHY, S. L.; KOCHANEK, K. D.; XU, J. Deaths: final data for 2012. 2015. Citado na página 19.

OBSTETRICIANS, A. C. of; GYNECOLOGISTS et al. Newborn screening and the role of the obstetrician–gynecologist. committee opinion no. 616. *Obstet Gynecol*, v. 125, p. 256–260, 2015. Citado na página 19.

OLAH, C. Understanding lstm networks. *GITHUB blog, posted on August*, v. 27, p. 2015, 2015. Citado na página 31.

PIERI, J. et al. Compact long-term recorder for the transabdominal foetal and maternal electrocardiogram. *Medical and Biological Engineering and Computing*, Springer, v. 39, n. 1, p. 118–125, 2001. Citado 2 vezes nas páginas 35 and 59.

RASU, R.; SUNDARAM, P. S.; SANTHIYAKUMARI, N. Fpga based non-invasive heart rate monitoring system for detecting abnormalities in fetal. In: IEEE. *Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on*. [S.l.], 2015. p. 375–379. Citado 4 vezes nas páginas 25, 34, 35, and 59.

RODRIGUES, J. A. Implementação da comunicação sem fio de um módulo estimador da frequência cardíaca fetal baseado em fpga. 2017. Citado na página 34.

ROSA, J. L. G. Biologically plausible artificial neural networks. In: *Artificial Neural Networks-Architectures and Applications*. [S.l.]: InTech, 2013. Citado na página 28.

SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2010. p. 92–101. Citado 2 vezes nas páginas 11 and 30.

TUTIDA, H. I. C. Implementação de um mødulo de aquisição de ecg abdominal em gestantes para estimativa da frequência cardíaca fetal usando fpga. 2017. Citado na página 34.

WARRICK, P.; HAMILTON, E.; MACIESZCZAK, M. Neural network based detection of fetal heart rate patterns. In: IEEE. *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. [S.l.], 2005. v. 4, p. 2400–2405. Citado na página 19.

YI, Z. *Convergence analysis of recurrent neural networks*. [S.l.]: Springer Science & Business Media, 2013. v. 13. Citado na página 31.